UNLIMITED

Report No. 90005

Report No. 90005

ROYAL SIGNALS AND RADAR ESTABLISHMENT, MALVERN

LATTICES FOR SECURITY POLICIES

Author: S N Foley (CITI, Milton Keynes)

PROCUREMENT EXECUTIVE, MINISTRY OF DEFENCE
RSRE
Malvern, Worcestershire.

April 1990

UNLIMITED

CONDITIONS OF RELEASE

0074113 BR-114204

************************* DRIC U

************************* DRIC Y

Reports quoted are not necessarily available to members of the public or to commercial organisations.

ROYAL SIGNALS AND RADAR ESTABLISHMENT

REPORT 90005

Title: **Lattices for Security Policies**

Author: Simon N. Foley[†]

Date: April 1990

Abstract

This report lays the foundation for a new model and approach for secure information flow. The model is driven by lattice based information flow policy, which describes the permitted dissemination of information in the system. System entities are allowed to handle different classes of information from the flow policy, and information is permitted to flow between entities so long as they do not violate the flow policy.

With this conceptually simple notion of security we can describe many interesting security policies, for example, traditional multi-level policies, aggregation policies, and chinese walls. Details are given on how secure systems based on the model can be implemented in practice. We also examine how other types of security policies such as integrity and separation of duty can be defined in terms of lattice based policies.

This report has been furnished for use by Her Majesty's Government under the terms and conditions of agreement 2241/02 from the Royal Signals and Radar Establishment.

[†] Cranfield Information Technology Institute
Pitfi-ld, Kiln Farm,
Milton Keynes, MK11 3LG

Contents

1 Introduction

Confidentiality security is concerned with restricting the disclosure of information in systems. One way of achieving this is to use an information flow policy which defines the different classes of information (for example, classified, secret, etc.) that can exist in the system and a flow relation which describes how information may flow between these classes. System entities (users, processes, files, etc.) are considered to be the sources and sinks of information, and each is bound to a security class from the flow policy. This binding is interpreted as: if entity A is bound to class a then A may source information of class a or higher and may sink information of class a or lower. A system is considered multilevel secure if all flows between entities maintain the flow policy. Note that there are special cases where certain entities, such as trusted subjects, are allowed violate this requirement in a controlled manner.

Entities may be statically or dynamically bound to their security classes. A statically bound entity has a fixed security class that cannot change during its lifetime. The class of a dynamically bound entity may change to accommodate incoming or outgoing data, or to allow direct upgrading or downgrading by some authorized entity. In the case of dynamic binding, a policy on how their classes may change is normally adopted. For example a high water mark policy[17] only allows classes to rise. With this approach, class changes cannot result in a violation of the multilevel security requirement. Some models permit controlled violation of the multilevel security requirement, such as McLean's[14] class change rules.

In this paper we propose two changes to this traditional notion of secure information flow. Firstly, each entity is bound to a group or set of security classes from the flow policy. This set defines the classes of information that the entity is allowed sink or source. Thus, an entity 'bound' to set {secret, top-secret} may sink and/or source secret or top-secret information. Of course, all flows will be subject to the constraint that they not violate the flow policy. With this change, we discover that we can describe a variety of new confidentiality policies, in particular aggregation policies. Secondly, we propose that flow policies need not always be based on a transitive flow relation. We give examples of useful non-transitive flow policies, and show how Chinese wall policies such as those in [5,12,15] can be constructed in terms of a reflexive flow policy and group bindings for entities. The advantage to our approach is that all these policies can be built within the framework of the same security model.

Section 2 proposes the new (abstract) model for secure information flow. Section 3 gives a refinement of the group confinement model, in the form of a state based mandatory access control model. This implementation oriented model is not unlike traditional MAC models, and we discuss how group confinement could be retrofitted to models such as Bell and LaPadula.

Information flow policies have traditionally been taken to be transitive[6], however in [10] a case is made for reflexive flow policies that may contain a non-transitive flow relation. Section 4 of this paper takes reflexive policies one step further by defining *reflexive lattices*—a reflexive policy with lowest upper and greatest lower bound operators that are useful for calculating the security class of aggregate information. We propose that a flow policy should form a reflexive lattice and show how an arbitrary reflexive relation can be transformed into a reflexive lattice (which in turn can be defined in terms of lattice operations facilitating its implementation). Section 5 shows how a selection of chinese wall policies can be described in terms of reflexive flow policies and group confinements.

A number of interpretations can be made about policies that are described as reflexive lattices. Common interpretations are: flow policies, which describe the allowable flows between security classes; integrity policies, which define the clearances necessary to affect the integrity of information. Section 6 proposes a new interpretation for a reflexive lattice that allows it to describe separation of duty policies.

The appendix contains the necessary proofs for all properties proposed in the paper.

2 Group Confinement Model

As with interval confinement in [10], we will define the concept of group confinement in as general terms as possible. If desired, the model can be subsequently refined along with the definition of information flow. For example, section 3 will consider a refinement to a state transition model.

Definition 1 The group confinement model (*GCFM*) is defined as

$$GCFM \doteq (ENTS, L, confine, \rhd)$$

where

- *ENTS* defines the set of entities of interest in the system. An entity is anything which information can flow into and/or out of, in the system. Examples are objects, files, and processes. If desired, even parts of the hardware can be considered as entities, for example, a terminal (source and sink of information to the peolpe who use it), a local area network, etc.
- L gives the information flow policy as a distributive lattice, with flow relation (partial order) $\leq$ [1], least upper and greatest lower bound operators $\vee$ and $\wedge$, respectively. We will study the information flow policy in more detail in section 4.
- $\rhd$ is an information flow relation between entities. The relation $E \rhd F$ implies that information can flow from entity E to entity F over the system being studied. It is generalized to $\mathcal{E} \rhd \mathcal{F}$, where $\mathcal{E}, \mathcal{F} \subseteq ENTS$, to mean information from the entities in $\mathcal{E}$ can flow as an aggregate, to the entities in $\mathcal{F}$. The relation $\rhd$ can be thought of as an abstraction of the implementation of the system into information flows between entities. How this relation is determined will depend on how the system is modeled, and the interpretation attached to the notion of information flow. In some cases, the flow relation might not reflect all the flows that could occur, rather it represents the flows to which the information flow policy L must apply.
- Every entity $E \in ENTS$ is confined to a set *confine*(E) (also denoted $\underline{E}$), of security classes from αL

$$Confine : ENTS \rightarrow Gclass; \quad (Gclass \doteq (\mathcal{P}\alpha L) - \{\})$$

[1] If L is a lattice then αL gives its set of components; $\overset{L}{\leq}$ its partial ordering; $\overset{L}{\vee}$ and $\overset{L}{\wedge}$ its lowest upper and greatest lower bound operators respectively. We will often drop the L from the operator when no ambiguity can arise. Similar abbreviations will be made for the other, yet to be defined, policy relations and operators.

where $\mathcal{P}\,A$ gives the powerset of the set A. The group confinement of an entity E is interpreted as:

- E is permitted to source information to any class s iff there exists an $a \in \underline{E}$ such that $a \leq s$.
- E is permitted to sink information from any class s iff there exists an $a \in \underline{E}$ such that $s \leq a$.

Note that all flows are subject to the constraint that they do not violate the flow policy—the *multilevel security requirement* must hold for confinement, i.e.,

information flows from class a to class b implies $a \overset{L}{\leq} b$

$\diamond$

Entities in *GCFM* can be thought of as bound to a *confinement group* from the *group policy* $L_{\mathcal{G}}$ with alphabet $\alpha L_{\mathcal{G}} = Gclass$. From the interpretation of group confinement, information from confinement group $A \in \alpha L_{\mathcal{G}}$ is information that could flow (sink) to any class s such that $a \leq s$, where $a \in A$, or could originate (source) from any class s such that $s \leq a$ and $a \in A$. This results in the following definition for the group flow relation $\leadsto$ on $L_{\mathcal{G}}$.

Definition 2 Information at a confinement group A may flow to confinement group B if $A \overset{L_{\mathcal{G}}}{\leadsto} B$ $(A, B \in \alpha L_{\mathcal{G}})$, where

$$A \overset{L_{\mathcal{G}}}{\leadsto} B \ \hat{=}\ \exists a \in A, b \in B \bullet a \overset{L}{\leq} b$$

Note that $\leadsto$ defined over $L_{\mathcal{G}}$ does not form a partial order because it is neither antisymmetric nor transitive. $\diamond$

Example 1 A system enforces a military style policy with classes u (unclassified), c (classified), s (secret), and t (top-secret). Entities A, B and C are confined to groups {u,c}, {c,s} and {s,t}, respectively. Information is permitted to flow from entity A to entity C since, entity A may source classified information and entity C may sink classified information. Information may not flow from C to A however, since the lowest classification of information that C can generate is secret, which cannot be sunk by A. Entity A is allowed source classified information and entity C allowed sink it so a flow from A to B is permitted.

Note that Entity B is also allowed source classified information, and entity A allowed sink this information, therefore a flow from B to A is allowed.

At first this may appear incorrect: what is preventing B sourcing secret information as classified and giving it to A? However, we view entities as entirely passsive, they generate information and the class of the information sourced at a particular instance will depend on the confinement of the entity, in addition to the destination of the information. Thus when B generates information destined for entity A is considerd to be classified. Similarly, when C generates information destined for B is is secret. Note that the system flow relation ($\rhd$) is not necessarily transitive, and thus $C \rhd B$ and $B \rhd A$ need not imply $C \rhd A$. Of course if it is transitive, then the scenario above is not secure since $C \rhd A$ is not secure.

Thus, suppose entities A and B are people, and we have an object D in the system confined to $\{s\}$ that is used to hold a secret. While entity B is permitted to transmit information to entity A, and is also allowed to read the secret in D, the entity (B) may not, by using the system, give the secret to entity A. For example, if B uses the mail utility to send a copy of D to A, then there is, in the system a flow $D \rhd A$, which is invalid. The entity B could always read the secret, and then re-type it and send it to user A as classified information. However in this case, the security breach is due to a flow outside the system (the 'human' copy), and therefore is not of interest for security in the system—the person could always use the telephone. If this type of flow is considered a threat, information flow through people can always be considered transitive so that $D \rhd B \wedge B \rhd A$ implies $D \rhd A$ always holds. Again, this reflects how the $\rhd$ relation represents the flows of interest over the system to which the information flow policy is to be applied. △

Another useful relation that can be defined over confinement groups is the bound order relation $\leq$. This relation determins if one confinement group forms a bound on another. We say that B is a bound on A iff every group that can flow to A can flow to B, and every group that B can flow to then A can also flow to. Formally,

Definition 3 Given confinement groups $A, B \in \alpha L_G$, define

$$\begin{aligned} A \overset{L_G}{\leq} B \quad &\doteq \quad \forall X \in \alpha L_G \bullet X \leadsto A \Rightarrow X \leadsto B \wedge \\ & \quad \forall X \in \alpha L_G \bullet B \leadsto X \Rightarrow A \leadsto X \end{aligned}$$

This is equivalent to,

$$A \overset{L_G}{\leq} B \Leftrightarrow \forall a \in A \bullet \exists b \in B \bullet a \leq b \wedge$$

$$\forall b \in B \bullet \exists a \in A \bullet a \leq b$$

The ordering $\leq$ forms a partial order over $L_{\mathcal{G}}$. ◇

If information at confinement group A is combined with information at group B then the confinement group of the resulting information should form an upper bound on both A and B, i.e, for every group that can flow ($\rightsquigarrow$) into A or B then it can also flow into their upper bound; simlarly, any group that the upper bound can flow into, then A and B can flow into.

Definition 4 Define the *upper aggregate* of confinement groups A and B from group policy $L_{\mathcal{G}}$ to be

$$A \overset{L_{\mathcal{G}}}{\oplus} B \doteq \{a \overset{L}{\vee} b | a \in A \wedge b \in B\}$$

Similarly, a *lower aggregate* can be defined as

$$A \overset{L_{\mathcal{G}}}{\otimes} B \doteq \{a \overset{L}{\wedge} b | a \in A \wedge b \in B\}$$

The upper aggregate of A and B forms an upper bound on A and B, and their lower aggregate forms a lower bound. They are also distributive. ◇

Example 2 An information flow policy for coordinates is defined in figure 1. A database of ICBMs is maintained which includes the longitude (group classification {`long`}), and latitude (group classification {`lat`}) of the location of each rocket. A system operator who is allowed to access either longitude or latitude information, but not both, is confined to group {`long`, `lat`}. Our definition of multilevel security ensures that this requirement is met, since an aggregate flow of longitude and latitude has confinement grouping {`coordinate`}, and the confinement of the operator excludes access to information in this group. △

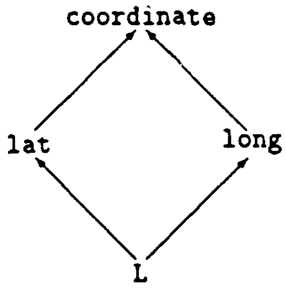


Figure 1: Coordinates Flow Policy

Unfortunately, while the upper aggregate is an upper bound operator, it does not give us a lowest upper bound. The following example will illustrate this, and also that the structure defined by $L_{\mathcal{G}}$ does not describe a lattice.

Example 3 Consider a group confinement policy constructed from the powerset lattice of the set $\{a,b,c\}$. This policy will have components $\{\{c\}\}$, $\{\{a\},\{b\}\}$, etc. Some upper aggregates are:

$$\{\{a\},\{b\}\} \oplus \{\{a\},\{b\}\} = \{\{a\},\{b\},\{ab\}\} \tag{1}$$

$$\{\{a\},\{b\}\} \oplus \{\{c\},\{b\}\} = \{\{ac\},\{b\},\{ab\},\{bc\}\} \tag{2}$$

$$\{\{a\},\{bc\}\} \oplus \{\{c\},\{ab\}\} = \{\{ac\},\{bc\},\{ab\},\{abc\}\} \tag{3}$$

Equation (1), illustrates that the upper aggregate of a group with itself may not result with the same group. Consider entities E and F bound to this group $\{\{a\},\{b\}\}$. The upper aggregate must consider the case where E can source information of class a and entity F can source information of class $\{b\}$; collectively they can source information of class $\{ab\}$ which is not in their original group confinements, but must be in their aggregate.

In equation (3), the operands A and B of the upper aggregate do not have a unique lowest upper bound. Two upper bounds of these components are:

$$\{\{ab\},\{bc\}\} \qquad \{ac\},\{abc\}\}$$

which are disjoint to one another and do not have a lower bound that forms an upper bound on A and B. Thus, $L_{\mathcal{G}}$ does not form a lattice. △

While $L_{\mathcal{G}}$ does not form a lattice, we know that it can be transformed into one using Denning's transformation (see section 4.1.1). However this transformation is unnecessary[2] as the aggregation operators have properties that can justify their use later in the paper. Given confinement groups A and B, then

$$\forall D \in \alpha L_{\mathcal{G}} \bullet (A \leq D \wedge B \leq D) \Rightarrow A \oplus B \leadsto D$$

i.e., the upper aggregate of A and B may flow ($\leadsto$) to any group that is a bound on A and B. A more general property is,

$$\forall D \in \alpha L_{\mathcal{G}} \bullet (A \leq D \wedge B \leq D) \Rightarrow \exists X \subseteq A \oplus B \bullet A \leq X \wedge B \leq X \wedge X \leq D$$

[2] and undesirable, since performing such a transformation would destroy the structure of $L_{\mathcal{G}}$ that makes it easy to implementusing bitwise operations on sets.

This equation can be thought of as reflecting the fact that while a pair of groups may not have a unique lowest upper bound, the aggregate will contain all candidates: for any upper bound D on A and B, a subset of the aggregate of A and B will form a lower bound on D.

The lower aggregate operator has similar properties: for $A, B \in \alpha L_\mathcal{G}$

$$\forall D \in \alpha L_\mathcal{G} \bullet (D \leq A \wedge D \leq B) \Rightarrow \exists X \subseteq A \otimes B \bullet X \leq A \wedge X \leq B \wedge D \leq X$$

Thus, while $A \otimes B$ does not give a greatest lower bound, (A and B may not even have a unique greatest lower bound), any lower bound of A and B will be dominated by some lower bound of A and B that is a subset of $A \otimes B$.

2.1 confinement Group Equivalence

There is a degree of redundancy in confinement groups. For example, the confinement groups {`classified`, `top-secret`} and {`classified`, `secret`, `top-secret`} from the military policy have the same meaning—information at any class may flow to these groups, while information at classified or higher may flow out.

Definition 5 Given a flow policy L, then for security classes $a, b, x \in \alpha L$, (a, b *covers* x) if a forms a lower bound on x and b forms an upper bound on x, i.e.,

$$a, b \textit{ covers } x \doteq a \overset{L}{\leq} x \wedge x \overset{L}{\leq} b$$

Extending this definition to a confinement group X covering a security class x gives

$$X \textit{ covers } x \doteq \exists a, b \in X \bullet a, b \textit{ covers } x$$

◇

Definition 6 Confinement groups A and B from group policy $L_\mathcal{G}$ are considered equal if $A \overset{L_\mathcal{G}}{\cong} B$, where

$$\begin{aligned} A \overset{L_\mathcal{G}}{\cong} B \doteq\ & \forall a \in A - B \bullet B \textit{ covers } a \wedge \\ & \forall b \in B - A \bullet A \textit{ covers } b \end{aligned}$$

◇

It can be shown that = forms an equivalence relation over $L_\mathcal{G}$, and that the following laws hold: for $A, B, X \in \alpha L_\mathcal{G}$ and if $A = X$ holds, then

$$\begin{array}{rclrcl} A \rightsquigarrow B & \Rightarrow & X \rightsquigarrow B & A \cup X & = & A \\ A \oplus B & = & X \oplus B & A \cap X & = & A \\ A \otimes B & = & X \otimes B & & & \end{array}$$

Since union and intersection over an equivalence class of confinement groups is closed, we can define the smallest and largest representitive classes of the equivalence class that group A is in, as

$$\begin{aligned} \lceil A \rceil &\hat{=} \cup\{X | X = A\} \\ \lfloor A \rfloor &\hat{=} \cap\{X | X = A\} \end{aligned}$$

In the examples throughout this paper, a confinement group A will normally be represented by the smallest component of its equivalence class, $\lfloor A \rfloor$. The normal set operators (union intersection etc.) can be defined over these groups as: for groups A and B, and class $a \in A$,

$$\begin{aligned} A \cup_{\mathcal{G}} B &= A \cup B & a \in_{\mathcal{G}} A &= a \in \lceil A \rceil \\ A \cap_{\mathcal{G}} B &= \lceil A \rceil \cup \lceil B \rceil & A \subseteq_{\mathcal{G}} B &= \lceil A \rceil \subseteq \lceil B \rceil \\ A -_{\mathcal{G}} B &= \lceil A \rceil - \lceil B \rceil \end{aligned}$$

To reduce the number of different operators, we will overload the normal set operators by dropping the $\mathcal{G}$ subscript in the group operators defined above.

2.2 Information Flow

The simple test[10] $E \rhd F \Rightarrow \underline{E} \ \le \ \underline{F}$ $(E, F \in \mathit{ENTS})$ is not a sufficient condition for multilevel security. The test does not consider the possibility of information from a number of entities flowing collectively into another entity. Since the aggregate operator for confinement groups does not give an upper bound, the individual flows may be valid (the confinement group of the destination entity dominates each individual source entity's confinement group), but the combined flow (aggregate of the confinement groups) may be invalid. Recall the security requirement—***information at confinement group A may flow to confinement group B iff $A \rightsquigarrow B$***; source information may have originated from a number of entities, and thus its confinement grouping is given by the upper aggregate of their confinement groups, and this information must be able to flow to a confinement group that is the lower aggregate of confinement groups of the destination entities. Thus, a system with group confinements is multilevel secure iff

$$\forall \mathcal{E}, \mathcal{F} \subseteq \mathit{ENTS} \bullet \mathcal{E} \rhd \mathcal{F} \Rightarrow \underline{\mathcal{E}}_{\oplus} \rightsquigarrow \underline{\mathcal{F}}_{\otimes} \tag{4}$$

where,

$$\underline{\mathcal{E}}_{\oplus} \ \hat{=} \ \oplus \{\underline{E} | E \in \mathcal{E}\} \quad \underline{\mathcal{F}}_{\otimes} \ \hat{=} \ \otimes \{\underline{F} | F \in \mathcal{F}\}$$

The group confinement model is not driven by a special aggregation policy. Rather, the information flow policy, in addition to defining the permitted flows in the system, also, inherently defines (possibly disjoint) classes and their aggregates (upper bounds). Group confinement allows one to distinguish the separate classes and their aggregates when binding.

Example 4 Continuing with example 2, the officer (entity O) has confinement

$$\underline{O} = \{\texttt{lat},\texttt{long}\}$$

and files A and B confinement,

$$\underline{A} = \{\texttt{lat}\} \quad \underline{B} = \{\texttt{long}\}$$

A system with flows $A \rhd O$ is secure since $\underline{A} \leadsto \underline{O}$. However, a system with flows $\{A, B\} \rhd O$ is not secure since $\underline{A} \oplus \underline{B} = \{\texttt{coord}\}$, which cannot flow to $\{\texttt{lat},\texttt{long}\}$. $\triangle$

Example 5 Entities E, F and X have confinements

$$\underline{E} = \{\texttt{a1},\texttt{a2}\} \quad \underline{F} = \{\mathsf{b}\} \quad \underline{X} = \{\texttt{x1},\texttt{x2}\}$$

drawn from the flow policy described in figure 2. A system with flows $\{E, X\} \rhd \{F\}$ and $\{X\} \rhd \{E\}$ is secure since X sources class $\texttt{x1}$ information to E ($X \rhd E$) and E and X sources $\texttt{a2} \vee \texttt{x2} = \mathsf{b}$ information to F. A system with flows $\{E, X\} \rhd \{E, F\}$ is not secure since

$$E \oplus X = \{\texttt{a1},\texttt{c},\texttt{d},\texttt{b}\} \not\leadsto \{\texttt{x3},\texttt{a2}\} = E \otimes F$$

This second scenario might correspond to a system where E receives information from X and then forwards that information with some of its own to F. The first scenario is the case where F receives independent information from E to X, and the information sourced by E is not the information it had sunk from X. This example shows the importance of representing (using $\rhd$) the nature of flows in the system correctly. $\triangle$

Note that this security requirement is not too strong, despite the fact that the aggregation operators, while providing upper and lower bounds, do not give lowest upper or greatest lower bounds: recall the properties for aggregation ($A, B \in \alpha L_{\mathcal{G}}$)

$$\forall D \bullet (A \leq D \wedge B \leq D) \Rightarrow A \oplus B \leadsto D$$
$$\forall D \bullet (D \leq A \wedge D \leq B) \Rightarrow D \leadsto A \otimes B$$

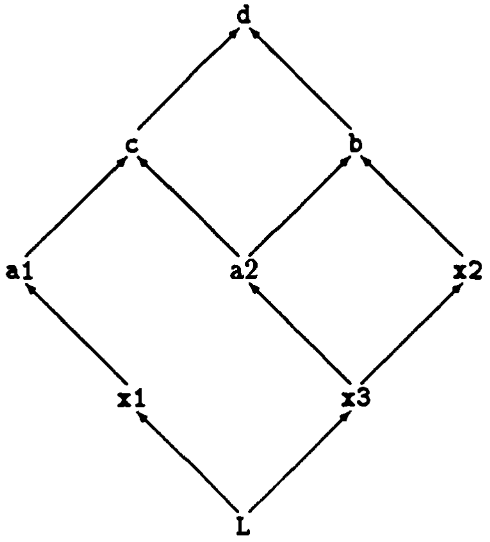


Figure 2: Flow policy

These imply that if there exists any upper bound D on the confinements of the entities of $\mathcal{E}$, that is also a lower bound on the confinements of the entities in $\mathcal{F}$, then $\underline{\mathcal{E}}_{\oplus} \leadsto D \leadsto \underline{\mathcal{F}}_{\otimes}$ holds. If such a bound does not exist the flow is not secure and $\underline{\mathcal{E}}_{\oplus} \leadsto \underline{\mathcal{F}}_{\otimes}$ does not hold.

Group confinement is a generalization of interval confinement[10]. Interval confinement is the case where each entity's group confinement forms an interval on the flow policy.

2.3 Group and Interval Confinement

In [10] entities are confined to an interval $[a, b]$ from the flow policy such that $a \leq b$. In this section we will show that the group confinement model is a generalization of the interval confinement model from [10].

Consider confinement groups of $L_{\mathcal{G}}$ that can be described as intervals of R, i.e., for $X \in \alpha L_{\mathcal{G}}$, then X can be described as an interval if

$$\exists a, b \in X \bullet \forall x \in X \bullet a \leq x \wedge x \leq b$$

If such an a, b exist it follows that $a \leq b$ and $\{a, b\} = X$. Now define an *interval policy* $L_{\mathcal{I}}$ whose components represent groups that can be described as intervals of R, i.e.,

$$\alpha L_{\mathcal{I}} = \{\{a, b\} | a \leq b \wedge a, b \in \alpha R\}$$

If $A = \{a, b\}$ is an element of $\alpha L_{\mathcal{I}}$ with $a \leq b$, then let A_L denote the lower bound a and A_H denote the upper bound b, on the interval described by A.

Now define for this policy $L_{\mathcal{I}}$ flow and aggregate operators based on those for $L_{\mathcal{G}}$ (i.e., definitions 2 and 4). This gives for $A, B \in \alpha L_{\mathcal{I}}$,

$$\begin{aligned} A \overset{L_{\mathcal{I}}}{\leadsto} B &\Leftrightarrow \exists a \in A, b \in B \bullet a \overset{R}{\leadsto} b \\ &\Leftrightarrow A_L \overset{R}{\leadsto} B_H \\ A \overset{L_{\mathcal{I}}}{\oplus} B &= \{A_L \cup B_L, A_H \cup B_H\} \\ A \overset{L_{\mathcal{I}}}{\otimes} B &= \{A_L \cap B_L, A_H \cap B_H\} \end{aligned}$$

We can show that $L_{\mathcal{I}}$ forms a lattice with an ordering relation defined as $(A, B \in \alpha L_{\mathcal{I}})$

$$A \overset{L_{\mathcal{I}}}{\leq} B \Leftrightarrow A_L \overset{R}{\leq} B_L \wedge A_H \overset{R}{\leq} B_H$$

Thus, $A \overset{L_{\mathcal{I}}}{\oplus} B$ gives a lowest upper bound and $A \overset{L_{\mathcal{I}}}{\otimes} B$ a greatest lower bound, on A and B. Therefore, in a system where all group confinements can be described as intervals (components of $L_{\mathcal{I}}$), the security requirement given above (equation (1)) can be simplified to

$$\forall E, F \in ENTS \bullet E \rhd F \Rightarrow \underline{E} \overset{L_{\mathcal{I}}}{\leadsto} \underline{F} \qquad (5)$$

In [10] entities are bound to intervals identical to those in $L_{\mathcal{I}}$, and the security requirement is

$$\forall E, F \in ENTS \bullet E \rhd F \Rightarrow \underline{E_L} \overset{R}{\leadsto} \underline{F_H}$$

which is equivalent to the requirement (2) above, given the definition of $\leadsto$ on $L_{\mathcal{I}}$. Therefore, interval confinement is a special case of group confinement where the groups can be described as intervals.

3 A State Machine Model

The group confinement model can be refined to a state machine model. In this section we will give one possible refinement, outlining how a restricted notion of group confinement can be implemented in a State based Mandatory access control Model (SMM).

The system model is represented by: an information flow policy L which forms a distributive lattice; a fixed set of system entities $ENTS$(subjects and objects); a set of system states S; an initial state s_0, and a state transition function T.

The system state in the MAC model describes the current accesses between subjects and objects and their current group confinements. For simplicity we choose to abstract the information about a system state to the following

- a function $confine(s, E)$ which returns the group confinement of entity $E \in ENTS$ at state s. This can also be written as $s.\underline{E}$.
- a transitive and reflexive relation $A \overset{s}{\triangleright} B$ which reflects the potential flows that could occur during state s due to the system accesses defined at state s.

The state transition function $T(op, s)$ returns a state s' which reflects the effect of applying some access change operation (op) at state s.

We will need to make a number of restrictions on this state model for our refinement to be correct. These will simplify the presentation of the model and also make it easy to implement in practice. Section 3.5 will consider the implications of these restrictions, and how they might me handeled in more general refinements of the GCFM. These restrictions are

- The relation $\overset{s}{\triangleright}$ is transitive and reflexive at any given state, and also transitive in the sense that if $A \overset{s}{\triangleright} B$ at state s, and $B \overset{s'}{\triangleright} C$ at a later state s', then there is a flow from A to C over the history of the system (see definition 7 for a formal definition of this).
- The confinement group of an entity contains an element that forms a lower bound on every component of its confinement, i.e., for $E \in ENTS$ then $\nabla(\underline{E})$ holds where, for confinement group A,

$$\nabla(A) \mathrel{\hat{=}} \exists l \in A \bullet \forall a \in A \bullet l \leq a$$

This component is denoted by $\perp_A$—the lowest bound on confinement group A (note that it exists and is unique since the original policy L forms a lattice).

We will now give an informal development of security for the SMM. Section 3.5 proves that the SMM is a refinement of GCFM.

If an entity has a group confinement A at state s then the information held by that entity can be thought of as having some classification $a \in A$. If there is a potential for a flow $E \overset{s}{\rhd} F$ during this state then this flow is valid so long as every possible class that can be sunk by F could have originated from E, i.e.,

$$\forall f \in s.\underline{F} \bullet \exists e \in s.\underline{E} \bullet e \leq f \tag{6}$$

This is in a sense like a high water mark. An entity confined to $\{\texttt{classified},\texttt{topsecret}\}$ must change its confinement to $\{\texttt{secret},\texttt{topsecret}\}$, if a read from a secret file is to be secure (preventing the entity forwarding the information as classified). The confinement of each entity must adhere to the restriction for group confinement, i.e., it must contain a lowest bound on all its components. Thus a state s is secure only if

$$\forall E \in ENTS \bullet \bigtriangledown(s.\underline{E})$$

This, along with equation (6) gives the condition for a secure state: state s is secure iff

$$(\forall E, F \in ENTS \bullet E \overset{s}{\rhd} F \Rightarrow \forall f \in s.\underline{F} \bullet \perp_{s.\underline{E}} \leq f) \wedge \forall E \in ENTS \bullet \bigtriangledown(s.\underline{E})$$

(note that $s. \perp_{s.\underline{E}}$ is dominated by every e of $s.\underline{E}$, and hence the simplification of (6).)

The confinement group of an entity may change as the system progresses so long as the changes do not result in a violation of the multilevel security requirement. In making a transition from state s to state s', an entity should not be allowed source or sink information at state s' that it could not source or sink at state s. Thus if

$$\{x \in \alpha L | \exists y \in s.\underline{E} \wedge y \leq x\}$$

gives the security classes from which entity E may source information at state S, then at state s'

$$\forall x \in s'.\underline{E} \bullet \exists y \in s.\underline{E} \bullet y \leq x \tag{7}$$

must hold. Similarly for sinks,

$$\forall x \in s'.\underline{E} \bullet \exists y \in s.\underline{E} \bullet x \leq y \tag{8}$$

Combining (7) and (8) above gives gives the requirement for a secure transition: for all entities $E \in ENTS$

$$\forall x \in s'.\underline{E} \bullet s.\underline{E} \textit{ covers } x \tag{9}$$

which is equivalent to $s'.\underline{E} \subseteq s.\underline{E}$, where $\subseteq$ describes subset over the largest components of its operands equivalence classes (see section 2.1). The state transition function is secure if for each state s and operation op, then the transition to $T(op, s)$ is secure.

The basic security theorem may now be stated.

Theorem 1 A state based MAC system, as described above, is secure iff

- s_0 is secure;
- every state reachable from state s_0 is secure, and
- transition function T is secure.

PROOF Covered in section 3.5 □

Example 6 A telephone directory holds the names and numbers of individuals from the accounts, personnel, and sales departments of a company. Each entry in the directory is bound to a single class from the set {acc,pers,sale}, identifying the department that each individual belongs to. Each employee allowed access to the directory may obtain numbers from at most two departments. Thus we define an information flow policy for the telephone directory as the powerset lattice $2^{\{\mathtt{acc,pers,sale}\}}$. Each employee allowed access to the directory is bound to the confinement group {{},{acc,pers},{acc,sale},{pers,sale}} (note the inclusion of {} to ensure that the confinement has a lowest bound).

Consider a system with telephone book entries labeled A, P, and S, with bindings

$$\underline{A} = \{\mathtt{acc}\} \quad \underline{P} = \{\mathtt{pers}\} \quad \underline{S} = \{\mathtt{sale}\}$$

and an employee with an initial binding at state s_0 defined in table 1. This table describes a possible trace of the system. At state s_1 employee E accesses the directory for an accounts number. So that this access may be secure (i.e., a secure state), the confinement of E must change to $s_1.\underline{E}$ reflecting the fact that from this state, E may subsequently access information either of class **pers** or of class **sale**, but not both. Observe that the transition from s_0 to s_1 is secure since the binding of E at state s_0 covers its binding at state s_1. A similar (secure) change in the bindings of E is required for the transition to state s_2 to be secure. For all subsequent states (after s_2) E may access personnel and accounts numbers but may not access sales numbers. △

state s_i	Accesses	Confinement ($s_i.\underline{E}$)
s_0	None	{{},{acc,pers},{acc,sale},{pers,sale}}
s_1	$A \overset{s_1}{\triangleright} E$	{{acc},{acc,pers},{acc,sale}}
s_2	$P \overset{s_2}{\triangleright} E$	{{acc,pers}}

Table 1: A secure access sequence

The above example shows how a quantity based aggregation policy can be described in terms of group confinement. We have not considered the possible problem of inference, where an employee might know enough accounts and personnel numbers to deduce something about sales telephone numbers. Such flows might be analyzed using information theoretic techniques, with the (inferred) flows modeled as part of the $\triangleright$ relation, and thus would not be of concern to the group confinement model at this level of detail.

3.1 Formal Basis for the State Transition Model

We have given an informal derivation of a state based version of the group confinement model. We will now prove that it is, in fact, a refinement of the GCFM. To do this we will set up an abstraction relation between the components of GCFM and SMM (state MAC model). We will then prove that the definition of security in SMM implies security in GCFM. This proof will provide the formal basis for the basic security theorem proposed earlier.

The SMM and GCFM share the same set of entities. In the SMM, the initial confinement of an entity corresponds to the confinement assigned to it in the GCFM, i.e., $\mathit{confine}(E) = s_0.\mathit{confine}(E)$. Thus an entity confined to {secret,top-secret} in the GCFM will have this group as initial confinement. The fact that its confinement will change as the states progress is an implementation issue. We will assume that there are no flows defined at the initial state. This will ensure that it is secure for the initial confinement.

In the GCFM, the relation $\triangleright$ represents the flows that can occur over every possible history of the system. In the state based model, the relation $\overset{s}{\triangleright}$ describes the flows that could occur due to accesses at state s. Information flow in the SMM is thought of as transitive in the sense that if there is a flow $E \overset{s}{\triangleright} F$ at state s, and a flow $F \overset{s'}{\triangleright} G$ at a later state s' then over the history of the system, there is a flow from E to G. Thus we define the flows that could have occured over a system as:

Definition 7 If Σ_n denotes a sequence of transitions, starting from the initial state s_0, and finishing at state s_n then the flows that can occur over this history is described by the relation $\overset{\Sigma_n}{\triangleright}$, where for $E, F \in ENTS$,

$$E \overset{\Sigma_n}{\triangleright} F \doteq \begin{cases} \exists G \in ENTS \bullet E \overset{\Sigma_{n-1}}{\triangleright} G \wedge G \overset{s_n}{\triangleright} F & \text{if } n > 0 \\ E \overset{s_0}{\triangleright} F & \text{otherwise} \end{cases}$$

The behaviour of a system can be characterised by Σ—the set of all possible transition sequences. All possible flows that could occur over the system are defined as ($E, F \in ENTS$, $\mathcal{E}, \mathcal{F} \in ENTS$),

$$\begin{aligned} E \overset{\Sigma}{\triangleright} F &\doteq \exists \Sigma_n \bullet E \overset{\Sigma_n}{\triangleright} F \\ \mathcal{E} \overset{\Sigma}{\triangleright} \mathcal{F} &\doteq \exists \Sigma_n \bullet \forall E \in \mathcal{E}, F \in \mathcal{F} \bullet E \overset{\Sigma_n}{\triangleright} F \end{aligned}$$

Observe from the last equation that an aggregate flow is not created if the flows occur from different histories. In our refinement $\triangleright$ can be thought of as the abstraction of the flows over the system, i.e., the relation $\overset{\Sigma}{\triangleright}$. $\diamond$

To prove that the SMM is a correct refinement of the abstract GCFM we must prove that any system secure by the SMM is secure by the abstract GCFM. A system is secure by the GCFM if

$$\forall \mathcal{E}, \mathcal{F} \subseteq ENTS \bullet \mathcal{E} \triangleright \mathcal{F} \Rightarrow \underline{\mathcal{E}_\oplus} \rightsquigarrow \underline{\mathcal{F}_\otimes}$$

where,

$$\underline{\mathcal{E}_\oplus} \doteq \oplus \{\underline{E} | E \in \mathcal{E}\} \quad \underline{\mathcal{F}_\otimes} \doteq \otimes \{\underline{F} | F \in \mathcal{F}\}$$

A system is secure by the SMM if every possible Σ_n is secure (i.e., the basic security theorem holds). Transition sequence Σ_n is secure if every state it visits is secure and every state transition it makes is secure. Thus, if

$$\forall \mathcal{E}, \mathcal{F} \subseteq ENTS \bullet \mathcal{E} \overset{\Sigma}{\triangleright} \mathcal{F} \Rightarrow s_0.\underline{\mathcal{E}_\oplus} \rightsquigarrow s_0.\underline{\mathcal{F}_\otimes}$$

holds for a secure SMM system, then it is also secure by the GCFM ($s_0.\underline{\mathcal{E}_\oplus}$ is the upper aggregates on the initial confinements of entities in $\mathcal{E}$ and similarly $s_0.\underline{\mathcal{F}_\otimes}$ for lower aggregates). This is proven by theorem 2 in the appendix. Thus SMM is a refinement of SMM—every secure SMM system is a secure GCFM system.

Pre-condition	$\forall E \in ENTS \bullet$ $\quad \oplus \{s.\underline{X} \mid X \overset{op}{\rhd} E \rightsquigarrow s.\underline{E}\} \cap s.\underline{E}$
Post-condition	$\forall E \in ENTS \bullet$ $\quad s'.\underline{E} = \oplus \{s.\underline{X} \mid X \overset{op}{\rhd} E\}$ $\forall E1, E2 \in ENTS \bullet$ $\quad E1 \overset{s'}{\rhd} E2 \Leftrightarrow E1 \overset{op}{\rhd} E2$

Table 2: Transition function $Req(op, s)$

3.2 Precision of the State Machine Model

We must address the precision of the security model SMM: is it possible for a system that is not secure by SMM to be secure by GCFM? Clearly a machine modelled by SMM that sets all entity confinements to singleton sets on its first transition (to a secure state) will loose precision, in that subsequent secure flows might not be permitted. For example, an entity confined to {`classified`,`top-secret`}, on reading a secret entity could have its confinement set to {`secret`}; the entity can no longer be granted a read to a top-secret entity, while in the GCFM it should be permitted since {`secret`} $\oplus$ {`top-secret`} $\rightsquigarrow$ {`class`,`top`}.

Therefore we would like that any implementation of SMM would ensure that the shrinkages of group confinements during state transitions are minimal. To achieve this we will propose an implementation for the state transition function $T(op, s)$. This new function will firstly determine whether a particular access request is possible, and if so, define how resulting group confinements should be calculated. Furthermore, we will prove that the SMM based on this transition function is precise, i.e., any system not secure by SMM given this transition function is not secure by the GCFM[3].

Definition 8 Table 2 defines the transition function $Req(op, s)$ which, given state s, returns state s', reflecting the effect of applying access request op at state s. In this table, $\overset{op}{\rhd}$ gives the access flows that could result if operation op was granted. If the precondition does not hold, the operation is ignored, and state s remains unchanged. ◇

Example 7 Return to example 2. If the employee E requests a read to an accounts phone number, then we have {`acc`} $\rightsquigarrow s_0.\underline{E}$, i.e., the pre-condition on the request holds, and thus

[3] Remember that we are dealing with a restricted form of the GCFM, and thus the 'precision' is limited to this class of system

it is secure. At state s_1, the confinement of E is calculated as

$$\begin{aligned} s_1.\underline{E} &= \{\texttt{acc}\} \oplus s_0.\underline{E} \cap s_0.\underline{E} \\ &= \{\{\texttt{acc}\}, \{\texttt{acc,pers}\}, \{\texttt{acc,sale}\}, \{\texttt{acc,pers,sale}\}\} \cap s_0.\underline{E} \\ &= \{\{\texttt{acc}\}, \{\texttt{acc,pers}\}, \{\texttt{acc,sale}\}\} \end{aligned}$$

The request to then read a personell number is valid since $\{\texttt{pers}\} \rightsquigarrow s_1.\underline{E}$, and at state s_2 E's confinement is calculated as,

$$\begin{aligned} s_2 &= \{\texttt{pers}\} \oplus s_1.\underline{E} \cap s_1.\underline{E} \\ &= \{\{\texttt{acc,pers}\}\} \end{aligned}$$

Now, a request to read a sale's number is refused since $\{\texttt{sale}\} \not\rightsquigarrow s_2.\underline{E}$. $\triangle$

Definition 8 was arrived at by the result of lemma 19: a secure transition from secure state s to secure state s' with flows described by $\overset{op}{\triangleright}$ at state s' is possible iff

$$\forall E \in ENTS \bullet \oplus \{s.\underline{X} | X \overset{op}{\triangleright} E\} \rightsquigarrow s.\underline{E}$$

and furthermore, that the resulting state defined by $Req(op, s)$ is secure. Therefore, any transition sequence Σ_n of SMM, built up in terms of $Req(op, s)$, will only visit secure states s_i and only make secure transitions s_{i-1} to s_i $(1 \leq i \leq n)$ along the way. Thus any system having a transition function implemented by $Req(op, s)$ will be secure by SMM, and hence secure by GCFM.

Transition function $Req(op, s)$ provides the basis for a precise implementation of GCFM. Theorem 3 proves that for any transition sequence Σ_n, built according to function $Req(op, s)$ (which we represent as Σ_n^R) if a subsequent request op fails (at state s_n) its precondition then had the transition to state s_{n+1} (with flows described by $\overset{op}{\triangleright}$) been allowed, the resulting flows of Σ_{n+1} would not be secure by the GCFM. Formally, given state s_n reached by Σ_n^R,

$$\begin{aligned} &\exists E \in ENTS \bullet \oplus \{s_n.\underline{X} | X \overset{s_{n+1}}{\triangleright} E\} \not\rightsquigarrow s_n.\underline{E} \Rightarrow \\ &\quad \neg(\forall \mathcal{E}, \mathcal{F} \subseteq ENTS \bullet \mathcal{E} \overset{\Sigma_{n+1}}{\triangleright} \mathcal{F} \Rightarrow s_0.\underline{\mathcal{E}}_{\oplus} \rightsquigarrow s_0.\underline{\mathcal{F}}_{\otimes}) \end{aligned}$$

Thus any request for a new access that fails the precondition on function Req correctly does so.

3.3 SMM and Traditional State Models

There is a similarity between the SMM and traditional state models. The SMM binds each entity to an element of a partial order relation (the group confinement policy); the notion of a secure state is defined exclusively in terms of this relation, and there is a rule based on this relation that defines how entity binding may change as the system progresses. While the SMM 'flow policy' ($L_{\mathcal{G}}$) does not form a lattice, it is partially ordered and does provide a consistent and meaningful treatment of upper and lower aggregate operators.

The SMM can implement a form of high water mark policy: if an entity is bound to an interval (i.e., contains unique upper and lower bounds on all components), then the test for a secure state becomes

$$E \overset{s}{\rhd} F \Rightarrow \bot_{s.\underline{E}} \leq \bot_{s.\underline{F}}$$

A state transition s to s' is secure if

$$\forall E \in ENTS \bullet \bot_{s.\underline{E}} \leq \bot_{s'.\underline{E}}$$

i.e., its lowest bound can only rise. It is unlike a high water mark in that the lowest bound may not rise beyond the top of the confinement interval. Thus an entity confined to {classified, secret} cannot sink top-secret information. A true high water mark policy can be achieved if the top of the interval is the top of the lattice flow policy, allowing bottoms rise to the top.

There is a useful class of systems described by a Bell and LaPadula model[1] that can be viewed as restricted instansiations of the SMM model. Take a BLP system where the flow model is described by a group confinement policy $L_{\mathcal{G}}$; subjects and objects (entities) are bound to single elements (groups) of this policy; the definition of a secure state (ss-condition and *-property) remain the same, and can be abstracted to: state s is secure iff

$$\forall E, F \in ENTS \bullet E \overset{s}{\rhd} F \Rightarrow s.\underline{E} \overset{L_{\mathcal{G}}}{\leq} s.\underline{F} \tag{10}$$

where $\leq$ is the (partial) bound order defined over the policy $L_{\mathcal{G}}$. The above condition for a secure state is stronger than the definition of a secure state in SMM since, by definition we have

$$\begin{aligned} A \leq B \;\Leftrightarrow\; & \forall a \in A \bullet \exists b \in B \bullet a \leq b \,\wedge \\ & \forall b \in B \bullet \exists a \in A \bullet a \leq b \end{aligned}$$

Finally, we have a restricted form of tranquility, whereby security classes (groups) may only change according to the rule for a secure state transition (equation (9)). Now we have a BLP-like model that can enforce aggregation policies.

Note that we have not given any consideration to how the policy $L_{\mathcal{G}}$ will be implemented. The confinement group equality relation and, in particular, lemma 6 implies that a confinement group can be represented by the smallest set in its equivalence class. This set can be shown to correspond to a set of disjoint intervals from the original flow policy, each interval covering the classes contained in it. In the case of the SMM, these intervals will have a common lowest bound—the lowest bound of the group. Entities are bound to lists of intervals, and as the system progresses and accesses are established, the intervals will shrink or be removed altogether to reflect the possible classifications of the information held by the entity.

3.4 Denial of Access

MAC models that incorporate dynamic binding can suffer from the problem of information flow due to denial of access: a low user can read a low file, however a high (Trojan) process, by writing high information into the dynamically bound file, prevents the low user from reading the file, and thus there is a potential for information flow. With SMM, a low file would be confined to $\{low\}$, and thus could not have high information written to it. To duplicate the scenario above, the file would have to be confined to $\{low, high\}$. Now the low user can initially read from this file. As soon as the high Trojan writes to the file, it is removed from the purview of the low user.

We are interested in how group confinement could be included in the Terry-Wiseman model of security[16]. In doing this, we must address how flows due to denial of access could be handelled under their philosophy for modelling security. Returning to the above scenario, the low user cannot create the file to be used, since any entity created by a person should inherit the confinement of the person, i.e., the file would be bound to $\{low\}$. The Trojan cannot create the file to be used since it is an untrusted piece of software and thus would not possess the faithful role. Thus to establish the covert channel, a faithful entity (person) confined to $\{low, high\}$ would have to create the file, and then make it available to both the Trojan and the low user.

There seem to be a number of ways to prevent this channel. The easiest approach is to prevent entity's changing their confinement when off the trusted path. This reflects the fact that while on the path, entities are trusted not to use class changes to transmit information; once off the path, the entity cannot be trusted, and thus security classes are not allowed change. Thus in the example above, the high entity decides what class of information (low or high) he wants put into the file (and updates its confinement appropriately) before he calls the Trojan horse.

Alternatively, we could insist that any class changes to an entity can only be done before the entity enters any subset of the public domain. Once an entity is made available to others, its confinement becomes static and can no longer change. This is the approach adopted in[11].

If the file is given to the low user or to the Trojan horse by the high user, then by the *no-signalling* rule there is no threat of flow since the requester (the high user) is trusted not to leak by using changes of control information. However, the high user is giving to the low user an entity that can be used to to establish a covert channel. While the high user can be trusted not create a channel based on controls from high to low, a low user cannot, and thus the low user should not be given an entity with confinement $\{low, high\}$. The low user may however be given the file confined to $\{low\}$ or $\{high\}$. This requirement needs to be formally captured in the *no-signalling* rule.

There is another type of flow due to denial of access that is not captured by any of the above approaches. Example 6 illustrated how a quantity based aggregation policy could be described in terms of group confinement. The security mechanism did not consider the possible problem of inferences that can be made from denial of access. An employee having accessed accounts and personell information might deduce that Jones is a member of the sales department if access to his/her number is denied. Thus if inference controls are to be included, they must consider how inferences can be made, not only about functional information in the system, but also about control information (confinement etc).

Information flows due to denial of access is a area for future work.

3.5 Other State Based Refinements of GCFM

The SMM is one, allbeit restricted, refinement of the group confinement model. It is restricted both in the kind of confinement it can enforce ($\nabla()$ must hold) and also in its interpretation of information flow (transitive and reflexive). However, we have seen, and will see in section 5 that it can enforce many useful security policies. This section considers how these restrictions on SMM might be avoided. Rather than proposing any concrete refinements, we will discuss possibilities for other refinements, some of which, we hope to develop formally in the near future.

3.5.1 Information Flow

Consider the defintion of information flow ($\overset{\Sigma}{\rhd}$) in SMM. This relation is transitive in the sense that information that flows into an entity can be extracted from that entity at a later state. This is a reasonable assumption for system objects such as files—information sunk to a file is generally done so that it can be retrieved later. In the SMM entities can be thought of as *memorable*—information that flows out of an entity could have been sunk by that entity during an earlier state.

However, this reasoning need not apply to people. A person who is confined to group {secret, top-secret} is trusted to handel secret and top-secret information appropriately (by the very fact that he/she is confined to this group). Such an person is trusted not to source top-secret information as secret, whether that top-secret information originated from within the system, or as some information from outside the system. This degree of trust can also be extended to any trusted process operating on the user's behalf. For example, if the window system software is certified and trusted to behave appropriately, then a user confined to {secret, top-secret} should be allowed to read and write simulaneously to a secret and top-secret windows on the terminal screen. The user is trusted not to read the top-secret window and re-type its information into the secret window. Note however, that the user should not be permitted to use any system software to, for example, cut a portion of the top-secret window and paste it into the secret window. In this latter case, an information flow is generated in the system as a result of the functionality of the windowing software and the flow policy must be applied. Therefore, our notion of trust does not permit arbitrary downgrades, unless the user is willing to manually re-type the top-secret information as secret, and in which case it would probably be easier to transmit the information using some other medium (for example, a telephone).

We view people as *memoryless*—information sourced by a person can be viewed as independant of any information sunk by that person during an earlier state. Recall the interpretation of the flow relation $\rhd$ in GCFM, as representing the flows to which the information flow policy is to be applied. In the case of memoryless people, once information leaves the system, it is assumed never to re-enter inappropriately, and thus the flow policy need not be applied to these flows. Note that while people may be considered memoryless in terms of other entities, they can remember information for their owm purposes: In example 2 the system operator is bound to {long,lat}. When he reads longitude information he can remember it at later states and thus must be prevented from reading latitude information. The operator is trusted to handel longitude or latitude information appropriately, but not both. If $ENTS_m$ returns the set of memorable entities from $ENTS$; $E \overset{s}{\rhd} F$ identifies the flows due to accesses at state s; then the flows as a result of a transition sequence Σ_n can be defined as

$$\begin{array}{lcll} E \overset{\Sigma_n}{\rhd} F & \hat{=} & \left\{ \begin{array}{ll} (\exists G \in ENTS_m \bullet E \overset{\Sigma_{n-1}}{\rhd} G \wedge G \overset{s_n}{\rhd} F) & \\ \vee E \overset{\Sigma_{n-1}}{\rhd} F \vee E \overset{s_n}{\rhd} F & \text{if } n > 0 \\ E \overset{s_0}{\rhd} F & \text{otherwise} \end{array} \right. \\ \mathcal{E} \overset{\Sigma}{\rhd} \mathcal{F} & \hat{=} & \exists \Sigma_n \bullet \forall E \in \mathcal{E}, F \in \mathcal{F} \bullet E \overset{\Sigma_n}{\rhd} F \end{array}$$

Note how the $(E \overset{\Sigma_{n-1}}{\rhd} F \vee E \overset{s_n}{\rhd} F)$ part reflects the fact that a memoryless entity can remember, for its own purposes, some information that could form part of a useful aggregate.

Given this definition, it is possible for two memoryless entities with similar confinements to cooperate and learn information that individually they should not have access to. Consider example 2, where there are two operators $O1$ and $O2$ each confined to {long,lat}. $O1$ is permited to read longitude information and forward it to $O2$ as latitude information, who in turn may read real latitiude information, resulting in access to a coordinate. Even if these entities were memorable, similar flows could be generated: $O1$ reads longitude, $O2$ reads latitude, and they compare values later in the canteen. Since the entities have the same confinement they can be expected to collude. This problem can be circumvented by the careful use of persistent-knowledge environments[15]. We could assume that if the operators are going to collude , they most likely share the same terminal, local network, or building. Thus we could assign and maintain, in the case of the terminal, a group confinement indicating the information that has been processed at the terminal. As accesses are made by its operators, its confinement will change. We do not need any special confinement mechanism for persistent knowledge environment labels, since a terminal can be treated as just another entity—a source and sink of information. We would have a similar treatment for people. A (memoryless) person confined to {long,lat} creates two windows on the screen. When the person reads longitude information through one window, the window's confinement will change to {long}, as well as the user's confinement (since there is a flow from the longitude information to the person). This prevents the user displaying latitude information in the other window. Note that if we are modelling the terminal as an entity, then its confinement would also change to {long}, preventing another person reading latitude information at that terminal.

Another class of memoryless entity is one where, even though it does 'forward' information using the system, it is trusted not to forward anything valuable. An example of such an entity is a (trusted) encryption process which sinks secret information (plain text) and sources unclassified information (cipher text). This process is certified to behave appropriately and thus can be considered memoryless. Again, this reflects the interpretation for $\rhd$ as representing the flows in the system to which the flow policy must be applied. Careful use of memoryless entities provide a form of controlled violation of the information flow policy—an unclassified user can only discover encrypted secrets. This idea is considered further in section 6

In this section we have only proposed the notion of memorable and memoryless entities, and the nature of the information flows between them. We have, as yet, to define the the necessary and sufficient conditions for a secure state and secure transition. This is one refinement of GCFM we believe to be worthwhile, and one we intend to do in the near future.

3.5.2 Confinement

The SMM can cater only for a restricted form of confinement: confinement groups must include an element that is a lower bound on the other members of the group. With this restriction in place SMM is a valid and precise refinement of GCFM.

If the restriction is removed, then SMM is no longer a valid refinement. This can be illustrated by the following example: a system has entities and confinements based on the flow policy from example 2,

$$\begin{array}{rclcrcl} \underline{E1} & = & \{\texttt{long,lat}\} & & \underline{A} & = & \{\texttt{long}\} \\ \underline{E2} & = & \{\texttt{long,lat}\} & & \underline{B} & = & \{\texttt{lat}\} \end{array}$$

The SMM can prevent entity $E1$ learning joint longitude and latitude information (i.e., a coordinate), but cannot prevent $E1$ sourcing *both* longitude and latitude information: consider the flows

$$E1 \overset{\#1}{\rhd} E2 \quad E2 \overset{\#2}{\rhd} A \quad E1 \overset{\#3}{\rhd} B$$

here, information from $E1$ flows to A and B, and while such a flow would be permitted in a SMM-like model, it is not allowed by the GCFM.

It was the above class of flow that lead us to adopt the restriction on group confinement. The restriction allows the security requirement to be re-written as

$$\forall \mathcal{E} \subseteq ENTS, F \in ENTS \bullet \mathcal{E} \rhd \{F\} \Rightarrow \underline{\mathcal{E}}_{\oplus} \leadsto \underline{F}$$

allowing us to ignore the calculation of lower aggregates for sinks , and thus avoids the problem noted above.

A valid state based model that does not have restrictions on confinement groups can be built by modifying the SMM appropriately. To capture flows such as the above (i.e., flows *to* aggregates) each entity E would require a history of the entities that originally sourced the information held by E. Given this, when the flow $E2 \rhd A$ occurs above, the confinement of $E1$ *and* $E2$ must change (to {long}) so that their aggregate sourced information can flow to A. The aim of such a mechanism would be to result in a property such as

$$\forall E, F \in ENTS \bullet E \overset{\Sigma_n}{\rhd} F \Rightarrow s_n.\underline{E} \leq s_n.\underline{F}$$

so that when the aggregates are calculated $(\mathcal{E} \overset{\Sigma_n}{\rhd} \mathcal{F})$ it would ensure that the security requirement of GCFM is upheld, since for groups A, B, C, D we have

$$(A \leq C \wedge A \leq D \wedge B \leq C \wedge B \leq D) \Rightarrow (A \oplus B \leadsto C \otimes D)$$

With this approach, we have a loss of precision due to the stronger definition of a secure state, and except possibly for the case of certification[8,13], it is not a practical approach since the history of all information flows would need to be maintained by the system at run-time.

An alternative to the above is to strengthen the notion of a secure state further so that state s is secure iff

$$\forall E, F \in ENTS \bullet E \overset{s}{\rhd} F \Rightarrow \forall e \in s.\underline{E}, f \in s.\underline{F} \Rightarrow e \leq f$$

With this restriction, once a flow is allowed during a state, it will be be permitted in all subsequent states since the entire confinement group of F strictly dominates every class of $s.\underline{E}$, and their shrinkage as the system progresses will have no affect on this ordering. Thus the property

$$\forall E, F \in ENTS \bullet E \overset{\Sigma_n}{\rhd} F \Rightarrow s_n.\underline{E} \leq s_n.\underline{F}$$

holds, and a system secure by the above will also be secure by GCFM. However, while the approach might initially appear attractive it is not at all precise and furthermore, unlike the SMM, the transition function does not lend itself to a clean implementation: flows between entities with confinement {long,lat} would require that their confinement change to *either* {lat} or {long}, imposing a commitment on the class of the flow betwen the entities before the nature of the flow (whether longitude or latitude) is known.

The SMM model, restricted as it is, does provide a framework for enforcing a variety of useful flow policies. We believe that the policies that it cannot enforce (i.e., $\nabla()$ does not hold for group cofinements) are either un-interesting (in the sense that we cannot find any useful application) or could be implemented as part of an integrity policy.

Consider the kinds of confinements that cannot be enforced by SMM. Such a confinement group C will have an $a, b \in C$ but $a \wedge b \not\in C$. Thus an entity E_C confined to C can source or sink information of class a or b, but cannot source or sink information of class $a \wedge b$. Now consider entities E_A, E_B confined to $A = \{a\}$ and $B = \{b\}$ respectively. Entity E_C can source to E_A or E_B, but not both. What would this sourcing correspond to in a system? If it corresponds to a direct write by entity E_C to entities E_A and E_B, then this n-person control is an integrity control and could be enforced as part of an integrity policy (see section 6). If the sourcing corresponds to a flow from a read, whether direct or indirect (for example a covert channel), then is the security control useful? We have something passive like a read of E_C by E_A at one state preventing a read of E_C by another entity (E_B) at a later state. Further work needs to be done to determine if such policies can be put to any use, and if so, how a pratical model to enforce them could be built.

4 Reflexive Flow Policies

Confidentiality security is concerned with restricting the disclosure of information in systems. One way of achieving this is to use an information flow policy[6] which can be thought of as defining the different classes or kinds of information that can exist in the system and how they may propagate. An example of this is the military flow policy which defines a set of security classes which represent the sensitivity of the information in the system, and an ordering relation which describes relative sensitivity. A flow relation ($\_ \leadsto \_$) defines how information may propagate and is interpreted as:

> for classes a, b, then $a \leadsto b$ means that information is permitted to flow from class a to class b.

Rather than thinking of these *classes* of information as just security classes (as in the traditional sense), we should think of them as a way of associating a *simple* representation of meaning with the information in the system. For example, a student computer system might have information of class exams, results, assignments, etc, and a flow relation which describes the allowable propagation of information between classes.

Formally, if R is an information flow policy, then αR is the set of security classes, and ($a, b \in \alpha R$) $a \overset{R}{\leadsto} b$ gives the flow relation. A flow relation is trivially reflexive, since information should always be allowed flow between the same security class. In [6], Denning argues that a flow relation should be transitive. However Foley[10] proposes a number of flow policies that use a non-transitive flow relation. In later sections (4.1 and 5) further examples of non-transitive flow policies will be given. Therefore, we will opt for the most general case where a flow relation need not be transitive.

Two security classes from an information flow policy can be considered equivalent if their flow relations with all other security classes are identical. We will make the reasonable assumption that a flow policy does not contain any equivalent security classes, i.e.,

$$\forall a, b \in \alpha R \bullet (\forall x \in \alpha R \bullet (x \leadsto a \Leftrightarrow x \leadsto b) \wedge (a \leadsto x \Leftrightarrow b \leadsto x)) \Rightarrow a = b$$

We call this requirement ***pseduo-antisymmetry***, because if the policy is transitive, then this requirement becomes one of antisymmetry.

Given that a flow policy R is reflexive and pseduo-antisymmetric, then the combination of information at class $a \in \alpha R$ with information at class $b \in \alpha R$ should result in information at a class $d \in \alpha R$ that forms an upper bound on a and b, i.e., $a \leadsto d \wedge b \leadsto d$. As R may not be transitive, d should also form an upper bound on all classes that are dominated by a or b, since information at class a or b may have originated from a lower class. Similarly, if the combined information (originally from classes a and b) at this class d may flow to some class c, then individual flows from a to c and from b to c must also hold. These observations give rise to the following definition.

Definition 9 A class d is an upper bound on a and b from the policy R if d *is-upper* a, b, where

$$\begin{aligned} d \textit{ is-upper } a, b \;&\hat{=}\; \forall c \in \alpha R \bullet c \leadsto a \vee c \leadsto b \Rightarrow c \leadsto d \wedge \\ &\phantom{\hat{=}}\;\; \forall c \in \alpha R \bullet d \leadsto c \Rightarrow a \leadsto c \wedge b \leadsto c \end{aligned}$$

A similar definition can be given for lower bounds. A class d is a lower bound on classes a and b from the reflexive policy R if d *is-lower* a, b, where

$$\begin{aligned} d \textit{ is-lower } a, b \;&\hat{=}\; \forall c \in \alpha R \bullet a \leadsto c \vee b \leadsto c \Rightarrow d \leadsto c \wedge \\ &\phantom{\hat{=}}\;\; \forall c \in \alpha R \bullet c \leadsto d \Rightarrow c \leadsto a \wedge c \leadsto b \end{aligned}$$

We can generalize these conditions in terms of a *bound order* relation: security class a is a lower bound of security class b in policy R if $a \overset{R}{\leq} b$, where

$$a \overset{R}{\leq} b \;\hat{=}\; a \textit{ is-lower } b, b$$

◇

The bound order relation of a policy is a partial order, and the following laws hold: for $a, b, c \in \alpha R$ then

$$\begin{aligned} a \leq b &\Rightarrow a \leadsto b \\ a \leq b &\Leftrightarrow b \textit{ is-upper } a, a \\ a \leq b \wedge a \leq c &\Leftrightarrow a \textit{ is-lower } b, c \end{aligned}$$

Note that we can rewrite the pseudo-antisymmetry requirement in terms of bound order as

$$\forall a, b \in \alpha R \bullet a \leq b \wedge b \leq a \Rightarrow a = b$$

Thus if policy R is antisymmetric under $\leq$ it is pseudo-antisymmetric under $\leadsto$.

A pair of security classes can have a number of upper bounds. The lowest of all these gives the lowest upper bound which provides us with the basis for security class combination.

Definition 10 A class d forms a lowest upper bound on classes a and b from policy R if

$$\forall d' \in \alpha R \bullet d' \textit{ is-upper } a, b \Rightarrow d \leq d'$$

A class d forms a greatest lower bound on a and b if

$$\forall d' \in \alpha R \bullet d' \textit{ is-lower } a, b \Rightarrow d' \leq d$$

Note that the antisymmetry of bound order ensures that d above is unique. ◇

Definition 11 An information flow policy forms a *reflexive lattice* if its flow relation is reflexive and pseduo-antisymmetric, and every pair of components have unique lowest upper, and greatest lower, bounds. ◇

Note that if R is a reflexive lattice then the set of security classes ordered by the bound order forms a lattice with bound operators defined by the existing bound operators of R.

4.1 Transforming Flow Policies

The group confinement model is defined in terms of a (partial order) lattice information flow policy. In this section we will show how an arbitrary reflexive relation can be transformed into a reflexive lattice, and how this can be enforced within the group confinement model. Before giving a general approach, we will first show how relations that are transitive and reflexive can be transformed into lattices.

4.1.1 Transforming Quosets into Lattices

In this section we will give three different approaches to transforming a quoset (reflexive and transitive relation) into a lattice. The first two are based on Birkoff[4] and Denning[7]. The third is a new transformation. All these transformations preserve the orderings in the original information flow policy, but treat existing upper and lower bounds from the quoset flow policy differently.

Birkoff Transformation

Definition 12 Given an arbitrary quoset Q, with ordering relation $\leq$, define a function

$$f^{\mathcal{P}}_Q : \alpha Q \rightarrow \mathcal{P} \alpha Q \quad f^{\mathcal{P}}_Q(a) \doteq \{b | b \overset{Q}{\leq} a\}$$

which maps Q to the powerset of αQ. We will drop the subscript from $f^{\mathcal{P}}_Q$ to give $f^{\mathcal{P}}$ if no ambiguity can arise. ◇

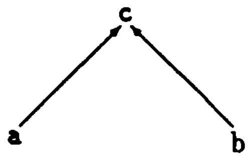


Figure 3: Quoset Q

The resulting lattice from the transformation is the powerset lattice $\mathcal{P}\,\alpha Q$ with ordering relation $\subseteq$, lowest upper and greatest lower bound operators $\cup$ and $\cap$ respectively. That the mapping $f^{\mathcal{P}}$ preserves the orderings defined by the quoset is proven in [4], i.e.,

$$\forall a, b \in \alpha Q \bullet a \overset{Q}{\leq} b \Leftrightarrow f^{\mathcal{P}}(a) \subseteq f^{\mathcal{P}}(b)$$

Note that there is a dual of this flow preserving mapping from αQ to $\mathcal{P}\,\alpha Q$ defined as

$$g^{\mathcal{P}}_{Q} : \alpha Q \rightarrow \mathcal{P}\,Q \quad g^{\mathcal{P}}_{Q}(a) \doteq \{b | a \overset{Q}{\leq} b\}$$

We will generally use the mapping $f^{\mathcal{P}}$.

Example 8 A quoset Q has alphabet $\{\mathsf{a}, \mathsf{b}, \mathsf{c}\}$ and orderings as defined in figure 3. This policy is mapped to the powerset lattice $\mathcal{P}\,\{\mathsf{abc}\}$ by the mapping

$$f^{\mathcal{P}}(\mathsf{a}) = \{\mathsf{a}\} \quad f^{\mathcal{P}}(\mathsf{b}) = \{\mathsf{b}\} \quad f^{\mathcal{P}}(\mathsf{c}) = \{\mathsf{abc}\}$$

Note how the orderings in Q are preserved by the mapping. For example, we have $\mathsf{a} \leq \mathsf{c}$ and $\{\mathsf{a}\} \subseteq \{\mathsf{abc}\}$. However, the 'intuitive' lowest upper bound of a and b is not $f^{\mathcal{P}}(\mathsf{c})$ in the lattice, but $\{\mathsf{ab}\} \subset f^{\mathcal{P}}(\mathsf{c})$.

Thus, if we were to use this technique to convert a quoset flow policy into a lattice for use in the GCFM, we must be aware of this 'side effect' where the apparent lowest upper and (greatest lower) bounds of the flow policy may not correspond to the lowest upper and (greatest lower) bounds in the lattice. Of course, in some instances this is is necessary if, in the original quoset, elements have more than one lowest upper bound. For example, suppose that both a and b above can also flow to an element d, then in Q, a and b do not have a unique lowest upper bound. However, in the powerset lattice, $\{\mathsf{ab}\}$ is their lowest upper bound, which in turn is dominated by both $f^{\mathcal{P}}(\mathsf{c})$ and $f^{\mathcal{P}}(\mathsf{d})$. △

An advantage of having our policy as a powerset lattice is that it can easily be implemented in terms of bitwise comparisons and operations on sets. There is, of course, a problem in implementing very large flow policies: each security class requires $|\alpha Policy|$ bits. In the group confinement model we are not storing a single security class for each entity, but a list of security classes. This list (at worst case) could range from a single component to $|\alpha Policy|$ components—the case where every class in the policy is disjoint (thus no advantage can be taken of class covering), and the entity is confined to the group $\{\{x\}|x \in \alpha Policy\}$. Perhaps large policies could be specified as a hierarchy of flow policies, each with their own orderings as well as orderings between individual policies. Such policies might find application in large networks, where nodes have local flow policies, in addition to net-wide policies.

Denning Transformation

Denning's transformation is similar to Birkoff's detailed above, however instead of mapping to the entire powerset lattice, the minimal subset of the powerset is taken, such that it includes the set of mapped components $f^P(x)$ $(x \in \alpha Q)$ along with necessary lowest upper and greatest lower bounds. The transformation is performed in two stages:

1. (Birkoff[4]) Transform the quasi ordered set Q into a partially ordered set (poset) P by

   (a) Define an *order preserving* mapping $f^P(x)$ from αQ to 2^Q as

   $$f^P(x) = \{y|y \in \alpha Q \wedge y \overset{Q}{\leq} x\}$$

   (b) Define poset P as

   $$\alpha P = \{f^P(x)|x \in \alpha Q\}$$

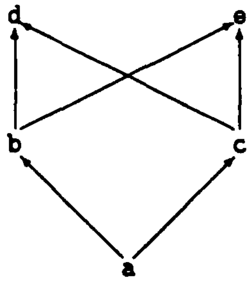
   and an ordering relation defined by subset i.e., for $X, Y \in \alpha P$ then

   $$X \overset{P}{\leq} Y \Leftrightarrow X \subseteq Y$$

   The mapping $f^P(x)$ from Q to P preserves the ordering relation[4] i.e.,

   $$\forall x, y(x, y \in \alpha Q \Rightarrow x \overset{Q}{\leq} y \Leftrightarrow f^P(x) \overset{P}{\leq} f^P(y))$$

2. (Denning[7]) The poset P can be transformed into a lattice L with: subset as the ordering relation, union as the lowest upper bound operator, and intersection as the greatest lower bound operator. The steps are:



x	$f^P(x)$
a	{a}
b	{a,b}
c	{a,c}
d	{a,b,c,d}
e	{a,b,c,e}

Figure 4: Quasi Ordered set Q, and mapping $f^P(x)$

(a) Add components $\mathtt{H} = \alpha Q$, and $\mathtt{L} = \{\}$ to αP.

(b) Define the set of successors common to sets of classes X and Y of poset P as $s(X,Y)$, where

$$s(X,Y) = \{Z | Z \in \alpha P \wedge X \cup Y \subseteq Z\}$$

(c) let $W = \bigcap s(X,Y)$. If $w \in s(X,Y)$, then class X and Y have class W as their lowest upper bound in P, otherwise they have none, so add W to αP.

It can be proven that this expansion of P terminates and that the orderings of the original policy are preserved by mapping $f^P(x)$.

Example 9 The policy of example 8 can be transformed into the lattice with components {}, {a}, {b} and {abc}. Here, the original lowest upper bound of a and b are preserved in the lattice as f^P(c). $\triangle$

Example 10 Figures 4 and 5 show the transformations from a quoset Q to a lattice L. $\triangle$

When we construct a flow policy we define all the different classes (kinds) of information that can exist in the system and define a flow relation over them. We may not wish to define what the precise upper and lower bounds are. In example 8, we define that information of class a may flow to c and that information of class b can also flow to c. Should this imply a commitment to having c as the lowest upper bound of a and b? We think not[4].

[4] Our argument in favour of Birkoff's transformation is applicable only to the work presented here, and in particular the group confinement model. There are situations where it *is* desirable to preserve as much of the structure of the original policy as possible, and Denning's transformation must be used. A case in point is the work in [15]

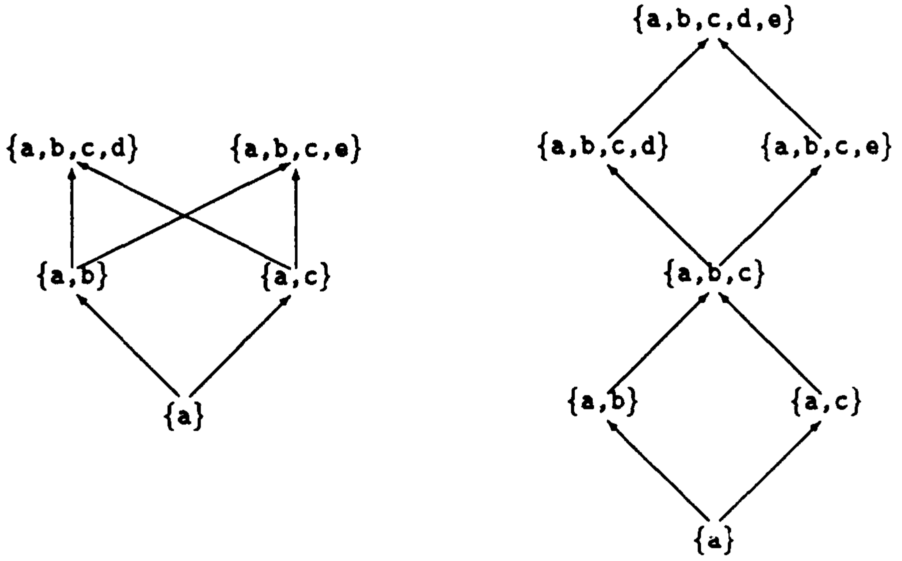


Figure 5: Poset P and Lattice L

Birkoff's transformation allows the lowest upper bound of a and b to denote a class that corresponds to the composition of information of class a and b (which may flow to c). With Denning's transformation we loose this fine granularity, and can only talk about the class c representing information at class c or the combination of information at classes a and b. If a large policy is to be built, then we feel that it is easier to think of the restrictions in terms of just a collection of flow relations (a may flow to b, etc.) than having to also consider aggregates and how they interact.

Example 11 An information system holds medical financial and personell details. The flow policy has an alphabet $\{\mathtt{med}, \mathtt{fin}, \mathtt{per}\}$, and each class is disjoint from one another. Using Denning's transformation, two additional classes will be added, namely H which denotes the universal upper bound, and L the universal lower bound. Given this basic policy, we cannot distinguish between information that originated from medical and financial sources or medical and personell sources. Consider an entity confined to the group

$$\{(\mathtt{med} \vee \mathtt{fin}), (\mathtt{med} \vee \mathtt{per})\}$$

which represents a person who is allowed read medical and financial *or* medical and personell information, but not everything. If we build the policy using Denning's transformation the confinement is implemented as

$$\{f^P(\mathtt{med}) \vee f^P(\mathtt{fin}), f^P(\mathtt{med}) \vee f^P(\mathtt{per})\} = \{\mathtt{H},\mathtt{H}\} = \{\mathtt{H}\}$$

which does not capture our intention. However, if we transform to the complete powerset lattice, then our desires are realized, with the entity bound to

$$\{f^{\mathcal{P}}(\mathtt{med}) \cup f^{\mathcal{P}}(\mathtt{fin}), f^{\mathcal{P}}(\mathtt{med}) \cup f^{\mathcal{P}}(\mathtt{per})\} = \{\{\mathtt{med},\mathtt{fin}\}, \{\mathtt{med},\mathtt{per}\}\}$$

Note that if we had started out with a powerset lattice (as was done in example 6) this problem would not have occurred. However, remember that we should think of a flow policy as defining the different kinds of information that can exist in the system (medical, financial, etc.), and how they may flow. The Birkoff transformation will incorporate how the aggregates may flow. △

Continuing with the last example, but considering Birkoff's transformation, the group confinement

$$\{(\mathtt{med} \wedge \mathtt{fin}), (\mathtt{med} \wedge \mathtt{per})\}$$

would be transformed to

$$\{f^{\mathcal{P}}(\mathtt{med}) \cap f^{\mathcal{P}}(\mathtt{fin}), f^{\mathcal{P}}(\mathtt{med}) \cap f^{\mathcal{P}}(\mathtt{per})\} = \{\{\}, \{\}\} = \{\{\}\}$$

loosing our desire to restrict flows *to* medical and financial, or medical and personell. Thus, while Birkoff's transformation provides a general treatment for the lowest upper bounds of classes from the original policy it is not consistent for greatest lower bounds. That is, given classes a and b, then $f^{\mathcal{P}}(a) \cup f^{\mathcal{P}}(b)$ is the upper bound on only a, b and anything they are bounds on; $f^{\mathcal{P}}(a) \cap f^{\mathcal{P}}(b)$ is a lower bound on a and b and anything they are bounded by, but may also be a lower bound on a class they are disjoint to. Put formally, given arbitrary quoset Q and mapping $f^{\mathcal{P}}$ to powerset lattice $\mathcal{P}\,\alpha Q$, then

$$\forall A, B \subseteq \alpha Q \bullet A \updownarrow B \Rightarrow (\cup\{f^{\mathcal{P}}(a) | a \in A\}) \updownarrow (\cup\{f^{\mathcal{P}}(b) | b \in B\}) \tag{11}$$

where $(a, b \in \alpha Q)$ $a \updownarrow b$ implies that classes a and b are disjoint, i.e.,

$$a \updownarrow b \doteq \neg a \leq b \wedge \neg b \leq a$$

this can be extended to sets of classes, such that given $A, B \subseteq \alpha Q$

$$A \updownarrow B \doteq \forall a \in A, \forall b \in B \bullet a \updownarrow b$$

Equation (11) tells us that given collections of classes A and B that are disjoint from one another, then their lowest upper bounds are also disjoint from one another. For example, {**fin**,**med**} and {**per**} are disjoint and so is their respective lowest upper bounds in the powerset lattice. However this law does not hold for greatest lower bounds, i.e.,

$$\forall A, B \subseteq \alpha Q \bullet A \updownarrow B \Rightarrow (\cap\{f^{\mathcal{P}}(a) | a \in A\}) \updownarrow (\cap\{f^{\mathcal{P}}(b) | b \in B\}) \tag{12}$$

does not hold for the powerset lattice as confirmed above[5].

The next section proposes a new transformation that provides a uniform treatment of greatest lower and lowest upper bounds.

Symmetric Powerset Lattices

Definition 13 Define a symmetric powerset lattice based on the set of elements S as $\mathcal{P}_S\, S$, with alphabet

$$\alpha\mathcal{P}_S\, S \;\hat{=}\; \{(X, Y) | X, Y \in \mathcal{P}\, S\}$$

If $A \in \alpha\mathcal{P}_S\, S$ then let (A_L, A_H) denote A. For $A, B \in \alpha\mathcal{P}_S\, S$ define

$$\begin{array}{rcl} A \leq B & \hat{=} & A_L \supseteq B_L \wedge A_H \subseteq B_H \\ A \vee B & \hat{=} & (A_L \cap B_L, A_H \cup B_H) \\ A \wedge B & \hat{=} & (A_L \cup B_L, A_H \cap B_H \end{array}$$

$\mathcal{P}_S\, S$ forms a distributive complementable lattice, with partial order $\leq$, and lowest upper and greatest lower bound operators $\vee$ and $\wedge$ respectively. ◇

Definition 14 Given an arbitrary quoset Q, define a flow preserving mapping $f^{\mathcal{P}_S}$ as

$$f_Q^{\mathcal{P}_S} : \alpha Q \rightarrow \alpha(\mathcal{P}_S\, \alpha Q) \quad f_Q^{\mathcal{P}_S}(x) = (g_Q^{\mathcal{P}}(x), f_Q^{\mathcal{P}}(x))$$

where functions $g^{\mathcal{P}}$ and $f^{\mathcal{P}}$ are defined in the previous section. ◇

The mapping $f^{\mathcal{P}_S}$ preserves the flows of it domain, i.e.,

$$\forall a, b \in \alpha Q \bullet a \overset{Q}{\leq} b \Leftrightarrow f^{\mathcal{P}_S}(a) \overset{\mathcal{P}_S\, \alpha Q}{\leq} f^{\mathcal{P}_S}(b)$$

Furthermore, if collections of classes are disjoint from one another then so are their lowest upper and greatest lower bounds, i.e.,

$$\begin{array}{rcl} \forall A, B \subseteq \alpha Q \bullet A \updownarrow B & \Rightarrow & (\vee\, \{f^{\mathcal{P}_S}(a) | a \in A\}) \updownarrow (\vee\, \{f^{\mathcal{P}_S}(b) | b \in B\}) \\ & & (\wedge\, \{f^{\mathcal{P}_S}(a) | a \in A\}) \updownarrow (\wedge\, \{f^{\mathcal{P}_S}(b) | b \in B\}) \end{array}$$

[5] **An interesting point is that if we use the dual mapping $g^{\mathcal{P}}$ then law (12) does hold while law (11) does not. This suggests how to construct the new transformation.**

Example 12 The flow policy described in example 11 can be transformed to the symmetric powerset lattice $\mathcal{P}_S\,\{\mathtt{med},\mathtt{fin},\mathtt{per}\}$ with mapping,

$$\begin{aligned} f^{\mathcal{P}_S}(\mathtt{med}) &= (\{\mathtt{med}\},\{\mathtt{med}\}) \\ f^{\mathcal{P}_S}(\mathtt{fin}) &= (\{\mathtt{fin}\},\{\mathtt{fin}\}) \\ f^{\mathcal{P}_S}(\mathtt{per}) &= (\{\mathtt{per}\},\{\mathtt{per}\}) \end{aligned}$$

Some (disjoint) greatest lower bounds are

$$\begin{aligned} f^{\mathcal{P}_S}(\mathtt{med}) \wedge f^{\mathcal{P}_S}(\mathtt{fin}) &= (\{\mathtt{med},\mathtt{fin}\},\{\}) \\ f^{\mathcal{P}_S}(\mathtt{med}) \wedge f^{\mathcal{P}_S}(\mathtt{per}) &= (\{\mathtt{med},\mathtt{per}\},\{\}) \end{aligned}$$

and entity confined to group $\{\mathtt{med} \wedge \mathtt{fin}, \mathtt{med} \wedge \mathtt{per}\}$ will be confined to

$$\{(\{\mathtt{med},\mathtt{fin}\},\{\}),(\{\mathtt{med},\mathtt{per}\},\{\})\}$$

in the model, and we get the desired flow controls. △

4.2 Transforming Reflexive Relations into Lattices

The group confinement model was defined in terms of a (partial order) lattice based flow policy. This section will show how an arbitrary reflexive relation can be transformed into a reflexive lattice, and enforced within the group confinement model. We will base our transformation on the powerset lattice. The appendix gives full details on how a reflexive policy can be mapped to a symmetric powerset lattice. For the sake of clarity we have choosen to present the less complex of the two approaches.

Given a reflexive flow policy R, construct a group confinement flow policy (section 2) built from the powerset lattice of the set of security classed defined by R. The alphabet of this lattice $R_{\mathcal{G}}$ is,

$$\alpha R_{\mathcal{G}} \doteq \{X | X \subseteq \mathcal{P}\,\alpha R\} - \{\}$$

and has a flow relation (defintion 2) defined as $(A, B \in \alpha R_{\mathcal{G}})$,

$$A \overset{R_{\mathcal{G}}}{\leadsto} B \Leftrightarrow \exists X \in A, Y \in B \bullet X \subseteq Y$$

(note that subset is the ordering relation on the 'base' policy αR for GCFM). This flow relation has a bound order defined by the partial ordering relation $\leq$ on $R_{\mathcal{G}}$ since by definition 3,

$$\begin{aligned} A \overset{R_{\mathcal{G}}}{\leq} B \;\Leftrightarrow\; & \forall X \bullet X \overset{R}{\rightsquigarrow} A \Rightarrow X \overset{R}{\rightsquigarrow} B \wedge \\ & \forall X \bullet B \overset{R}{\rightsquigarrow} X \Rightarrow A \overset{R}{\rightsquigarrow} X \end{aligned}$$

We know from section 2 that $R_{\mathcal{G}}$ does not form a lattice, but if we consider only group confinements that form intervals on $\mathcal{P}\,\alpha R$, i.e., for each confinement A,

$$\exists l, h \in A \bullet \forall x \in A \bullet l \leq X \wedge X \leq h$$

Then the set of all such confinements are closed over $\oplus$ and $\otimes$, and it forms a sublattice (an interval lattice[10] of $R_{\mathcal{G}}$ with $\oplus$ as lowest upper and $\otimes$ as greatest lower bound operators. Furthermore, from the above we know that this interval sublattice forms a reflexive lattice with flow relation $\rightsquigarrow$ and bound order $\leq$.

Definition 15 Define a mapping $f^{\mathcal{P}}_{R}$ from arbitrary reflexive relation R to this sublattice of $R_{\mathcal{G}}$ as

$$f^{\mathcal{P}}_{R} : \alpha R \rightarrow \alpha R_{\mathcal{G}} \qquad f^{\mathcal{P}}_{R}(a) \doteq \{l^{\mathcal{P}}_{R}(a), h^{\mathcal{P}}_{R}(a)\}$$

where

$$l^{\mathcal{P}}_{R}(a) \doteq \{b | b \overset{R}{\leq} a\} \quad h^{\mathcal{P}}_{R}(a) \doteq \{b | b \overset{R}{\rightsquigarrow} a\}$$

◇

Each component of R will be mapped to an interval of $\mathcal{P}\,\alpha R$, since bound is a stronger condition than flow, i.e., $a \leq b \Rightarrow a \rightsquigarrow b$, implying that $l^{\mathcal{P}}(a) \subseteq h^{\mathcal{P}}(a)$, and $f^{\mathcal{P}}_{R}(a)$ is an interval of $\mathcal{P}\,\alpha R$, with bottom $l^{\mathcal{P}}(a)$ and top $h^{\mathcal{P}}(a)$ Thus $f^{\mathcal{P}}$ is a mapping from R to a reflexive lattice. The mapping $f^{\mathcal{P}}$ preserves the flows of R, i.e.,

$$\forall a, b \in \alpha R \bullet a \overset{R}{\rightsquigarrow} b \Leftrightarrow f^{\mathcal{P}}(a) \overset{R_{\mathcal{G}}}{\rightsquigarrow} f^{\mathcal{P}}(b)$$

Thus we can use $f^{\mathcal{P}}(a)$ in $R_{\mathcal{G}}$ for any class a drawn from R with no detrimental effect on flows.

Since $R_{\mathcal{G}}$ has been built from a powerset lattice then, like the Birkoff transformation, it interval sublattice may contain extra upper bounds. For example, consider the flow policy in figure 3 (page 34), a,b and c from R would be mapped to {{a}}, {{b}} and {{a,b}} respectively in $R_{\mathcal{G}}$. The lowest upper bound of a and b in R is c, but in $R_{\mathcal{G}}$ it is {{a,b}}, reflecting the fact that information of class a and b have been combined. Refer back to section 4.1.1 for the pros and cons and this effect. Like the Birkoff transformation, $f^{\mathcal{P}}_R$ has the property,

$$\forall A, B \subseteq \alpha R \bullet A \updownarrow B \Rightarrow (\oplus \{f^{\mathcal{P}}_R(a) | a \in A\}) \updownarrow (\oplus \{f^{\mathcal{P}}_R(b) | b \in B\})$$

but need a transformation based on symmetric powerset lattices to achieve a similar property for greatest lower bounds.

Thus we have a transformation from an arbitrary reflexive policy R to a reflexive sublattice of $R_{\mathcal{G}}$ that preserves the flows of R and provides consistent treatment of upper and lower bounds.

Enforcing a Reflexive Policy in GCFM

Given a reflexive flow policy, the intention behind confining an entity E to a confinement group A of classes drawn from the policy, is that

1. E is permitted to source information to any class s iff there exists an $a \in A$ such that $a \overset{R}{\rightsquigarrow} s$.

2. E is permitted to sink information from any class s iff there exists an $a \in A$ such that $s \overset{R}{\rightsquigarrow} a$.

Consider condition 1. above. This can be written in terms of $R_{\mathcal{G}}$ based on the mapping $f^{\mathcal{P}}$ as: E is permitted to source information to any class $S \in \alpha R_{\mathcal{G}}$ iff

$$\begin{aligned}
& \exists a \in A \bullet f^{\mathcal{P}}(a) \overset{R_{\mathcal{G}}}{\rightsquigarrow} S \\
\Leftrightarrow\ & \exists a \in A \bullet \exists x \in f^{\mathcal{P}}(a), s \in S \bullet x \subseteq s \\
\Leftrightarrow\ & \exists x \in \cup\{f^{\mathcal{P}}(a) | a \in A\} \bullet x \subseteq s
\end{aligned}$$

If we consider only singleton sets $S = \{s\}$ drawn from $R_{\mathcal{G}}$ we get the condition

1. E is permitted to source information to any class $s \in \mathcal{P}\alpha R$ (the powerset lattice policy that drives $R_{\mathcal{G}}$) iff there exists an $a \in \cup\{f^{\mathcal{P}}(a) | a \in A\}$ such that $a \subseteq s$

We can simliarly re-define item 2. above using the same reasoning to give

2. E is permitted to sink information from any class $s \in \mathcal{P}\alpha R$ iff there exists an $a \in \cup\{f^{\mathcal{P}}(a) | a \in A\}$ such that $s \subseteq a$.

We now have a reformulation of what me mean by group confinement for reflexive policies in terms of a (transformed) group confinement based on classes drawn from a lattice flow policy. Thus we can enforce reflexive flow policies withing the existing structure of the GCFM.

Example 13 A private hospital information system processes information of class **records** (medical history); **treatment** (given to patients); **accounts** (for patients); **director** (shareholder information); and **management**. How information may flow between these different classes is described by the reflexive relation in figure 1. Note how **treatment** information is allowed flow to **records** or **management**, but for confidentiality reasons, cannot flow to class **director**. Similarly, **accounts** information is not allowed flow to **records** (for profitability reasons). Management is allowed coordinate all this information given these constraints. This reflexive relation can be transformed into the reflexive lattice $R_{\mathcal{G}}$ using the mapping described in table 3. Observe from this table that if information of class **treatment** and **accounts** are combined then their lowest upper bound in $R_{\mathcal{G}}$ is not f(**management**), but {{**t**,**a**}}, which may flow to class **management**, but not to **records** nor **director**.

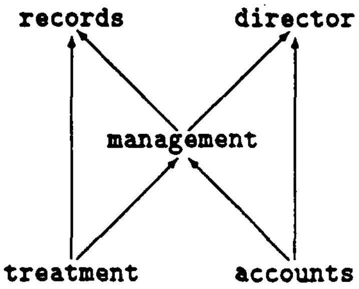


Figure 6: Flow policy HOSPITAL

x	$f(x)$
records	{{t,r},{t,m,r}}
director	{{d,a},{d,m,a}}
management	{{m},{t,m,a}}
treatment	{{t}}
accounts	{{a}}

Table 3: Mapping from HOSPITAL to $R_{\mathcal{G}}$

A consultant in this hosptial might be allowed sink and/or source information of class **treatment** and **records**, and thus is confined to the group $\{\texttt{treatment}, \texttt{records}\}$. In the GCFM this corresponds to a binding

$$\begin{aligned} f(\texttt{treatment}) \cup f(\texttt{records}) &= \{\{\texttt{t}\},\{\texttt{r}\},\{\texttt{tmr}\}\} \\ &= \{\{\texttt{t}\},\{\texttt{r}\},\{\texttt{tm}\},\{\texttt{tr}\},\{\texttt{rm}\},\{\texttt{tmr}\}\} \end{aligned}$$

The hospital administrator might be bound to $\{\texttt{management}\}$, which from table 1 corresponds to $\{\{\texttt{m}\},\{\texttt{tma}\}\}$. Note how the consultant and administrator are bound to groups of classes from the lattice $\mathcal{P}\alpha$HOSPITAL which is enforced by GCFM. These classes represent the sources/sinks that they are permitted to make. For example, class $\{\texttt{ma}\}$ means that the administrator can sink management and accounting information; the administrator can also source records information since $\{\texttt{m}\} \subseteq \{\texttt{tmr}\}$ is also in his confinement. Note that in the case of the MAC model described earlier, it is not allowed (and correctly so) do both: as soon as the administrator accesses accounts information, his confinement must change to $\{\{\texttt{ma}\},\{\texttt{tma}\}\}$ to ensure a secure state. Now it is no longer possible to source to records, since the administrator possesses accounts information which he has the ability to source, and **accounts** ⇝ **records** is invalid. Similarly, if the administrator initally sourced to records, he could no longer sink accounts. We shall see further examples of these kinds of aggregate policies in the next section. △

Thus, an arbitrary reflexive flow policy can be transformed and enforced by the GCFM. If the policy is transitive, then each component a of the policy αR will map to a singleton set $\{f(a)\}$ in $R_{\mathcal{G}}$, since $l(a) = h(a)$. If the policy is not transitive, then certain components of αR will map to a group of classes from $\mathcal{P}\alpha R$ (in fact an interval, since $l(a) \subseteq h(a)$). The mapping $f^{\mathcal{P}}$ from R to $R_{\mathcal{G}}$ applies to the abstract model GCFM. If we are to use a reflexive policy in the SMM, we must ensure that the resulting confinements are valid (i.e., $\nabla()$ holds). When a single class $a \in \alpha R$ is mapped to $f^{\mathcal{P}}(a)$, since $f^{\mathcal{P}}(a)$ forms an interval then $\nabla(f^{\mathcal{P}}(a))$ holds. The map of a group of classes $A \in \mathcal{P}\alpha R$ may not form a valid confinement for SMM. In this case, a lowest bound $l = \wedge \{\bot_{f^{\mathcal{P}}(a)} | a \in A\}$ should be added so that $\nabla(\{\bot_{f^{\mathcal{P}}(a)} | a \in A\} \cup \{l\})$ holds.

5 Chinese Wall Security Policies

The group confinement model can be used to formulate (and enforce) Chinese Wall security policies[5]. It has been pointed out that a chinese wall is a form of aggregation policy, where users are allowed access to individual datasets but not to aggregates of conflicting datasets. We will show, using a number of examples, how aggregation policies similar to those in [5,12,15] might be described within our model.

Example 14 Consider a market analysis database that contains information about banks Bank-x and Bank-y, and oil companies Oil-z and Oil-w. There are two conflict of interest classes `BANKS` and `OIL`. Within conflict class `BANKS` there are two kinds of information (datasets) `bank-x` and `bank-y`, corresponding to the class of information held by Bank-x and Bank-y respectively. The chinese wall policy insists that these classes are disjoint, i.e., information about one bank is not allowed flow to another bank. Thus the conflict of interest class `BANKS` can be thought of as describing a flow policy with alphabet $\{\texttt{bank-x}, \texttt{bank-y}\}$ and relations $\texttt{bank-x} \leadsto \texttt{bank-x}$, $\texttt{bank-y} \leadsto \texttt{bank-y}$. Conflict policy `OIL` has a similar definition, with alphabet $\{\texttt{oil-z}, \texttt{oil-w}\}$ and classes `oil-z` and `oil-w` disjoint.

Now we must define how these two policies can be composed. Flows are possible between the components of the conflict policies so long as they do not violate the relations within them. Therefore, the overall flow policy can be described by the join[10] of `BANKS` and `OIL`. Policy join $\sqcup$ of policies $C1$ and $C2$ has alphabet $(\alpha C1 \cup \alpha C2)$ and its flow relation is defined as $(a, b \in \alpha(C1 \sqcup C2))$,

$$\begin{aligned} a \overset{C1 \sqcup C2}{\leadsto} b \quad &\hat{=} \quad (a, b \in \alpha C1 \Rightarrow a \overset{C1}{\leadsto} b) \wedge \\ & \quad\quad (a, b \in \alpha C2 \Rightarrow a \overset{C2}{\leadsto} b) \end{aligned}$$

Table 4 gives the mapping of this policy to the group lattice built from the powerset lattice $\mathcal{P}\{\texttt{x}, \texttt{y}, \texttt{z}, \texttt{w}\}$, where `bank-x` is abbreviated to `x`, and similarly for the other classes. However, on a closer inspection of policy $\texttt{BANKS} \sqcup \texttt{OIL}$ we discover that it is too general for our purposes here: it (correctly by its definition) permits flows from `bank-x` to `oil-z` and from `bank-y` to `oil-z` and also from $\texttt{bank-x} \oplus \texttt{bank-y}$ to `oil-z`, which is not desirable. Thus, we need to supplement the defintion of join with a definition of how the aggregates of classes may flow in the joined policy. In the case of `oil-z` we wish to prevent it from

s	$f(s)$
bank-x	$\{\{\mathtt{x}\},\{\mathtt{xzw}\}\}$
bank-y	$\{\{\mathtt{y}\},\{\mathtt{yzw}\}\}$
oil-z	$\{\{\mathtt{z}\},\{\mathtt{xyz}\}\}$
oil-w	$\{\{\mathtt{w}\},\{\mathtt{xyw}\}\}$

Table 4: Mapping for BANKS $\sqcup$ OIL

sourcing/sinking aggregate bank-x $\oplus$ bank-y. Therefore we define class oil-z' as[6],

$$\begin{aligned} \mathtt{oil\text{-}z'} &= f(\mathtt{oil\text{-}z}) - (f(\mathtt{bank\text{-}x}) \oplus f(\mathtt{bank\text{-}y})) \\ &= \{\{\mathtt{z}\},\{\mathtt{xz}\},\{\mathtt{yz}\}\} \end{aligned}$$

Note how xyz is no longer a member of oil-z'. The other classes can be similarly redefined so as to constrain flows of (conflicting) aggregates,

$$\begin{aligned} \mathtt{bank\text{-}x'} &= f(\mathtt{bank\text{-}x}) - (f(\mathtt{oil\text{-}z}) \oplus f(\mathtt{oil\text{-}w})) \\ \mathtt{bank\text{-}y'} &= f(\mathtt{bank\text{-}y}) - (f(\mathtt{oil\text{-}z}) \oplus f(\mathtt{oil\text{-}w})) \\ \mathtt{oil\text{-}w'} &= f(\mathtt{oil\text{-}w}) - (f(\mathtt{bank\text{-}x}) \oplus f(\mathtt{bank\text{-}y})) \end{aligned}$$

Now, under this flow policy information is permited to flow between the classes of different conflict policies. For example, bank-x' and bank-y' may flow to oil-z', but their aggregate bank-x' $\oplus$ bank-y' may not.

Any information about Bank-x stored in the database will have group confinement bank-x'; information about Bank-y will have confinement bank-y', etc. A user of the database will be confined to (bank-x' $\cup$ bank-y' $\cup$ oil-z' $\cup$ oil-w'), allowing access to every individual item of company information but not to conflicting aggregates.

Consider a database with entries X, Y, Z and W confined as

$$\begin{aligned} \underline{X} &= \mathtt{bank\text{-}x'} & \underline{Z} &= \mathtt{oil\text{-}z'} \\ \underline{Y} &= \mathtt{bank\text{-}y'} & \underline{W} &= \mathtt{oil\text{-}w'} \end{aligned}$$

and a user E with confinement (bank-x' $\cup$ bank-y' $\cup$ oil-z' $\cup$ oil-w'). A system with flows $\{X, Z\} \rhd U$ is secure since

$$\underline{X} \oplus \underline{Z} \leadsto \underline{U}$$

[6] Note that the set difference operator is defined on the largest groups in the equivalence classes of its operands.

$$\Leftrightarrow \{\{\mathbf{xz}\},\{\mathbf{xyzw}\}\} \leadsto \{\{\mathbf{x}\},\{\mathbf{y}\}.\{\mathbf{z}\},\{\mathbf{w}\},\{\mathbf{xz}\},\{\mathbf{xw}\},\{\mathbf{yz}\},\{\mathbf{yw}\}\}$$

However, a system with flow $\{X,Y\} \rhd U$ is not secure since

$$\underline{X} \oplus \underline{Y} = \{\{\mathbf{xy}\},\{\mathbf{xyz}\},\{\mathbf{xyw}\}\}$$

may not flow to $\underline{U}$ since there is no component of $\underline{U}$ that contains $(\mathbf{xy})$. △

Although our example only considered the information flow in terms of the abstract relation $\rhd$, it applies equally to the MAC state based example: As the user accesses the database entries, his confinement will change to reflect the information he possesses, which, in turn, will ensure a valid chinese wall.

Lin[12] points out that in the Brewer-Nash model, conflict of interest satisfies a kind of transitivity property: if A is in conflict with B, and B in conflict with C, then A is (implicitly) in conflict with C. The GCFM does not impose this type of restriction as will be seen in the next example taken from [12].

Example 15 A (slightly outdated) database holds strategic information about the countries: USA, UK and USSR. Information classes **usa** and **uk** are disjoint to information of class **ussr** (i.e., information is not permitted to flow between them). Thus we define two conflict of interest classes C1 and C2, with alphabets

$$\alpha\mathtt{C1} = \{\mathtt{ussr},\mathtt{usa}\} \quad \alpha\mathtt{C2} = \{\mathtt{ussr},\mathtt{uk}\}$$

and no relations defined on them except reflexivity. We can construct $\mathtt{C1} \sqcup \mathtt{C2}$ using the same approach as in the last example. Remember that with this joined policy, all relations are valid so long as the relations of the original policies are preserved. This will yield a mapping from classes **ussr**, **usa** and **uk** to confinement groups defined as

$$\begin{aligned} f(\mathbf{usa}) &= \{\{\mathbf{usa}\},\{\mathbf{usa},\mathbf{uk}\}\} \\ f(\mathbf{uk}) &= \{\{\mathbf{uk}\},\{\mathbf{usa},\mathbf{uk}\} \\ f(\mathbf{ussr}) &= \{\{\mathbf{ussr}\}\} \end{aligned}$$

In this policy, information may flow between the USA and UK, but always remains disjoint from the USSR. △

In [15] Meadows considers how the Brewer-Nash Chinese wall can be extended so that it can handle military (multilevel) aggregation problems. For example, (conflicting) datasets

x	$f(x)$
f(a) (usa)	{ {a}, {akust} }
f(r) (ussr)	{ {r}, {rust} }
f(k) (uk)	{ {k}, {akust} }
f(u)	{ {u}, {akru} }
f(s)	{ {s}, {akrus} }
f(t)	{ {t}, {akrust} }

Table 5: Flow preserving mapping from MILAGG

A and B might be unclassified and secret repectively, however their aggregate is not secret, but top-secret. The next example considers how such a policy might be constructed in terms of a reflexive flow policy and group confinements.

Example 16 Consider example 15, and suppose we wish to introduce military-style classifications for the different kinds of information (datasets) in the system. With this policy, information of kind usa and/or uk is considered unclassified; information of kind ussr is secret; information formed from the aggregation of ussr and uk is secret; however, the aggregation of ussr and usa information is considered top-secret (an 'excepted aggregate'). Thus for example, a secret user of the database may read USSR and UK entries, but may only read either USSR *or* UK entries. A top-secret user may read all kinds of information. Note that all these flows are (implicitly) constrained by the flow policy; thus while the top secret user may initially be granted complete access to USA and USSR files, he may not copy information from one to the other.

We must construct a flow policy which defines the allowable flows between the different kinds (classes) of information. We start off with the usual partially ordered military policy MILITARY with alphabet $\{u, s, t\}$, where u denotes class secret; s denotes secret, and t top-secret. Given policies C1 and C2 from example 15, define the policy

$$\mathtt{MILAGG} \doteq \mathtt{C1} \sqcup \mathtt{C2} \sqcup \mathtt{MILITARY}$$

Table 5 gives the inital mapping f from MILAGG to its group policy (a abbreviates usa; k abbreviates uk, and r ussr). As with the last two examples we know that policy join is too general for aggregation puproses, so some additional constraints need to be placed on each class. Each dataset is constrained to a single security class from MILITARY; for example, datasets usa and uk are unclassified. Thus we have,

$$\begin{aligned} \mathtt{a'} &= f(\mathtt{a}) \cap f(\mathtt{u}) = \{\{\mathtt{au}\}, \{\mathtt{auk}\}\} \\ \mathtt{r'} &= f(\mathtt{r}) \cap f(\mathtt{s}) = \{\{\mathtt{rs}\}, \{\mathtt{rus}\}\} \\ \mathtt{k'} &= f(\mathtt{k}) \cap f(\mathtt{u}) = \{\{\mathtt{ku}\}, \{\mathtt{auk}\}\} \end{aligned}$$

The security classes from MILITARY are sufficient except that we need to ensure that the aggregate of USA and USSR information is not secret, but top-secret. This yields

$$\mathtt{s'} = f(\mathtt{s}) - (f(\mathtt{a}) \oplus f(\mathtt{r})) = \{\{\mathtt{s}\}, \{\mathtt{akus}\}, \{\mathtt{rkus}\}\}$$

A user of the database who is bound to secret (i.e., s'), can access any individual dataset, or the aggregate of USSR and UK. However, he may not access the aggregate of USSR and USA.

Again remember that these bindings can be used in the state based model described in section 3. In the case of the secret user, as soon as he accesses USSR information, he cannot write this information into a UK or USA location (preserve flow policy). Having done this, he cannot access any USA information (chinese wall). However, he can still *read UK information, since its aggregate with* USSR is secret. △

These last three examples illustrate some of the versatility of describing a security policy in terms of a reflexive flow policy and group confinement. We believe that there are many more interesting policies that can be described within the framework of the GCFM. However, it is not initially apparent how to construct the different policies in terms of reflexive policies and group confinements. For example, we had difficulty trying to capture the example policy described in [15] and for the sake of clarity had to resort to the simpler one described in example 16. We have a policy construction operator join $\sqcup$ which does not quite fit our requirements for composing aggregate policies. Further research is required to define additional, more useful, policy join operators. Ultimately, we envisage a flow policy description language, which can succinctly capture the flow and aggregation rules for a mandatory confidentiality policy. We believe that group policy lattices are sufficiently flexible to provide the basis for the semantics of such a language.

6 Security Flavours

The group confinement model provides an approach to modeling confidentiality requirements in systems. An information flow policy specifies the permitted dissemination of information as a flow relation between different classes (or kinds) of information that can exist in the system. Entities are bound to confinement groups, which provides a means of controlling the movement of aggregate information through the system.

6.1 Collective Confidentiality

A different interpretation can be made about an information flow policy. Given a reflexive lattice R, then $a \leadsto b$ $(a, b \in \alpha R)$ could be taken to mean that information at class a is permitted to flow to the *collective* classes $b_1, \ldots, b_n$, where $b = b_1 \vee \cdots \vee b_n$. This is not to state that the information may flow from a to the individual b_i's. In general, we say that for $a_1, \cdots, a_n, b_1, \ldots, b_m$ then $a_1 \wedge \cdots \wedge a_n \leadsto b_1 \vee \cdots \vee b_m$ means that information at the collective classes $a_1, \ldots, a_n$ may flow to the collective classes $b_1, \ldots, b_m$. We call a policy with such an interpretation on its relation a *collective flow policy*.

Example 17 A system contains a file of passwords of class **pass**. No single user may arbitrarily view this file, however a system manager (class **s-mgr**) and a system operator (class **s-op**) are allowed access to the file if they do so together (collectively). This collective policy is described by the reflexive relation in figure 7. In this policy, information is not allowed flow from class **pass** to a class **s-op**, or from **pass** to **s-mgr**. However, it is allowed flow from **pass** to a combination of system manager and operator, i.e., **pass** $\leadsto$ **s-mgr** $\vee$ **s-op**. $\triangle$

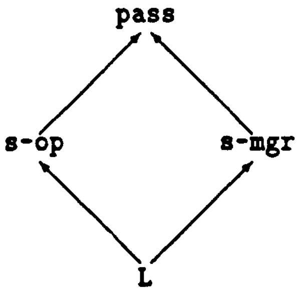


Figure 7: Collective flow policy

If entities are bound to classes drawn from a collective policy R, then a system enforces the policy R if

$$\forall \mathcal{E},\mathcal{F} \subseteq ENTS \bullet \mathcal{E} \rhd \mathcal{F} \Rightarrow \bigwedge_{E\in\mathcal{E}} \underline{E} \overset{R}{\rightsquigarrow} \bigvee_{F\in\mathcal{F}} \underline{F} \tag{13}$$

This requires that if $\mathcal{E} \rhd \mathcal{F}$, then the entities of $\mathcal{E}$ must have a collective classification that may flow to the collective classification of the entities of $\mathcal{F}$. Note that a similar requirement can be given for group confinements, where each entity is bound to a set of classes from the collective policy.

Example 18 Continuing with example 17, a password file, and users *Jones* and *Smith* have bindings

$$\underline{Pfile} = \texttt{pass} \quad \underline{Jones} = \texttt{s-op} \quad \underline{Smith} = \texttt{s-mgr}$$

A system with only the flow $Pfile \rhd Jones$ is not secure by requirement (13). However, a system with the flow $Pfile \rhd \{Jones, Smith\}$ is secure, since $\underline{Pfile} \rightsquigarrow \underline{Smith} \vee \underline{Jones}$ $\triangle$

Example 18 shows that an n-person (flow) rule can be specified in terms of a collective flow policy. The policy transform from section 4.1 implies that an n-person rule can be described in terms of a lattice, and enforced using its relations and operations. However, since the security requirement for collective policies is different to that for flow policies, a collective policy cannot be directly supported within the framework of existing flow models. Thus for example, the MAC model proposed in section 3 could not, in its present form, enforce a collective policy. Further research is needed to determine how the MAC model can be modified so that it can enforce these policies.

6.2 Integrity

A well known interpretation for a reflexive lattice is an integrity policy[3]. An integrity policy describes a set of clearances and a relation between these clearances that defines their relative superiority. If R is a reflexive lattice describing an integrity policy then for $a, b \in \alpha R$, if $a \rightsquigarrow b$ the clearance b is higher or superior to, the clearance of a. If the relation $A \rhd B$ is interpreted as *entity A can, in some way, affect the integrity of entity B*, then if $A \rhd B$, A must have a higher clearance than B. Thus for $A, B \in ENTS$,

$$A \rhd B \Rightarrow \underline{B} \overset{Policy}{\rightsquigarrow} \underline{A}$$

where the clearance of entity E is denoted $\underline{E}$. If entities are bound to a group of clearances, with an interpretation similar to that of group confinement for flows, then aggregate clearances must be considered. The aggregate integrity requirement

$$\forall \mathcal{E}, \mathcal{F} \subseteq ENTS \bullet \mathcal{E} \rhd \mathcal{F} \Rightarrow \underline{\mathcal{E}}_{\oplus} \overset{R_{\mathcal{C}}}{\leadsto} \underline{\mathcal{F}}_{\otimes} \tag{14}$$

states that the entities in $\mathcal{E}$ must have sufficient *common* clearances to affect the integrity of the entities in $\mathcal{F}$.

Example 19 The company in example 6 might have an integrity policy described by the powerset lattice of categories {**acc**,**pers**,**sale**}. An employee with an integrity confinement of {{**acc**}{**sale**}{**pers**}} is permitted to change the phone numbers of any single department, but not the numbers of different departments. An employee with integrity {{**acc**,**pers**,**sale**}} is allowed change the numbers of all departments (assuming his/her confidentiality confinement allows such access). △

6.3 Collective Integrity

An integrity policy can also be interpreted as a collective integrity policy. Such a policy describes what collective clearances are sufficient to allow the integrity of an entity to be changed. The collective integrity policy requirement follows from (13) and (14) above, and is

$$\forall \mathcal{E}, \mathcal{F} \subseteq ENTS \bullet \mathcal{E} \rhd \mathcal{F} \Rightarrow \underline{\mathcal{E}}_{\otimes} \overset{R_{\mathcal{C}}}{\leadsto} \underline{\mathcal{F}}_{\oplus} \tag{15}$$

Example 20 An integrity policy (figure 8) describes the relationship between the security clearances: cheques (**cheque**); company secretary (**sec**), and managing director (**md**). The

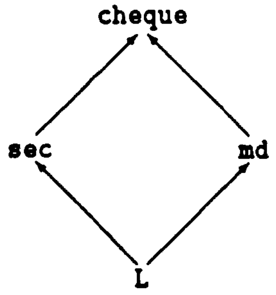


Figure 8: Collective flow policy

company secretary and managing director do not, individually, have sufficient clearance to write a cheque. However, their *collective* clearance can write a cheque. The company acountant may also be allowed write a cheque so long as he/she does so with the secretrary or the managing director. We do not want a commitment on the clearance for the accountant, and thus confine him/her to {**sec**,**md**}, providing the necessary aggregation control. Note that this clearance contains an assumption that if there are two accountants in the company, then they are trusted to write cheques jointly (but cannot write them individually). △

6.4 Controlled Violations

The notion of memoryless entities (section 3.5) together with reflexive flow policies provide an effective approach to controlled flow violations. Consider the private hospital from example 13 (page 43). Treatment information is not allowed flow to directors. A yearly report on treatment statistics might be required by directors, however the flow policy will not permit it. Introduce a new information class **t-stat** to the policy such that it is disjoint to all other classes except for the flows **treatmment** $\leadsto$ **t-stat** and **t-stat** $\leadsto$ **director**. This revised policy continues to ensure **treatment** $\not\leadsto$ **director**. Develop a (trojan free) process that scans treatment information and generates a statistical report. If we use statistical inference techniques to ensure that the report does not reveal anything anything about individual patients, then the reporting entity (the report process) can be regarded as memoryless, and confined to class {**t-stat**}. Now the only information that can flow from treatment to directory is treatment statistics[7].

Carefull use of this technique can provide an approach to typing information flow—the only information that can flow from class a to class b is information of class c. Of course, the validity of the technique rests with the ability to provide sufficient assurance that the entities confined to class c can be trusted to handle the information appropriately. The technique can also be used for integrity policies, and provides a method for ensuring that the integrity of an entity can be changed only by invoking a certain class of procedure. These procedures are trusted in the sense that they will alter the entity's integrity appropriately.

[7] assuming that all entities bound to class **t-stat** are appropriate statistical processes.

7 Conclusion

This report proposes a new approach to describing controls on the dissemination of information in system. The approach is driven by a flow policy that describes the allowable flows between different classes (or kinds) of information that can exist in a system, and each system entity is bound to a group of these classes. Unlike traditional binding, group confinement provides control on the movement of aggregate information flowing into and out of an entity.

The report provides evidence (via examples) that useful non-transitive flow policies do exist, and thus we call for a basic requirement that , flow relation can be any reflexive relation on a set of classes. We know from section 4 that such a relation can be transformed into a reflexive lattice, which in turn is defined in terms of lattice operations, thus facilitating its implementation. With this combination of reflexive flow policies and confinement groups complex policies can be described. To illustrate this, a selection of chinese wall policies were described.

Section 3 detailed how group confinement could be refined to a state based security model. The resulting model is not at all unlike existing state based models. Entities are bound to components from a flow policy (the group confinement policy), a inductive notion of secure state and secure transition can be captured and indeed a system could be built around a reference monitor if desired. As a first attempt at refining the GCFM, section 3 proposed a simple (yet effective) state based model. For the future we intend to develop more complex model refinements, hopefully culminating in a refinement that will be analogous to the Terry Wisemann security model in [16] enforcing group confinement.

Section 6 briefly considers how reflexive lattices can be used to specify properties other than confinement. The most obvious applications of reflexive lattices are confidentiality and integrity policies. The notion of a collective policy was introduced, where a reflexive lattice describes how collections of different classes of information may flow. Further work needs to be done to determine how collective policies can be enforced by state based models.

To conclude, this report has shown that it is possible to describe useful confidentiality, integrity, separation of duty, and aggregation, based policies in terms of lattices.

8 Acknowledgements

I would like to thank Jeremy Jacob for helpful comments on an earlier draft of this paper. Work on the group confinement model started while I was at the Department of Computer Science, University College, Cork. Since October 1989 the research has been funded by the Royal Signals and Radar Establishment. I would like to express my gratitude to them, and in particular Simon Wiseman and Peter Bottomley.

A Theorems and Lemmas

A.1 Group Confinement Model

Lemma 1 The bound order relation (definition 3) can also be defined as: for groups $A, B \in \alpha L_{\mathcal{G}}$,

$$\begin{aligned} A \leq B \;\Leftrightarrow\; & (\forall a \in A \bullet \exists b \in B \bullet a \leq b) \wedge \\ & (\forall b \in B \bullet \exists a \in A \bullet a \leq b) \end{aligned}$$

PROOF I By definition we have,

$$A \leq B \Rightarrow \forall X \in \alpha L_{\mathcal{G}} \bullet X \rightsquigarrow A \Rightarrow X \rightsquigarrow B$$

Pick singleton set X's from above, drawn from the set A, this gives,

$$A \leq B \Rightarrow \forall x \in A \bullet \{x\} \rightsquigarrow A \Rightarrow \{x\} \rightsquigarrow B$$

But $\{x\} \rightsquigarrow A$ always holds since $x \in A$. Thus,

$$\begin{aligned} A \leq B \;&\Rightarrow\; \forall x \in A \bullet \{x\} \rightsquigarrow B \\ &\Rightarrow\; \forall a \in A \bullet \exists b \in B \bullet a \leq b \end{aligned} \tag{16}$$

Similarly,

$$A \leq B \Rightarrow \forall X \in \alpha L_{\mathcal{G}} \bullet B \rightsquigarrow X \Rightarrow A \rightsquigarrow X$$

and picking singleton X's drawn from B gives,

$$\begin{aligned} A \leq B \;&\Rightarrow\; \forall x \in B \bullet B \rightsquigarrow \{x\} \Rightarrow A \rightsquigarrow \{x\} \\ &\Rightarrow\; \forall b \in B \bullet \exists a \in A \bullet a \leq b \end{aligned} \tag{17}$$

combining (16) and (17) gives

$$\begin{aligned} A \leq B \;\Rightarrow\; & \forall a \in A \bullet \exists b \in B \bullet a \leq b \wedge \\ & \forall b \in B \bullet \exists a \in A \bullet a \leq b \end{aligned}$$

PROOF II We have by definition, for $A, B, X \in \alpha L_{\mathcal{G}}$

$$B \rightsquigarrow X \Leftrightarrow \exists b \in B, x \in X \bullet b \leq x$$

but if $(\forall a \in A \bullet \exists b \in B \bullet a \leq b)$ holds, then there is an a such that $a \leq b$. and by transitivity $a \leq x$. Thus,

$$(\forall a \in A \bullet \exists b \in B \bullet a \leq b) \Rightarrow \forall X \bullet B \rightsquigarrow X \Rightarrow A \rightsquigarrow X \tag{18}$$

and similarly we can show,

$$(\forall b \in B \bullet \exists a \in A \bullet a \leq b) \Rightarrow \forall X \bullet X \rightsquigarrow A \Rightarrow X \rightsquigarrow B \tag{19}$$

and combining (18) and (19) gives

$$\begin{aligned} &(\forall a \in A \bullet \exists b \in B \bullet a \leq b) \\ &\wedge(\forall b \in B \bullet \exists a \in A \bullet a \leq b) \;\Rightarrow\; A \leq B \end{aligned}$$

□

Lemma 2 Equality (definition 6) forms an equivalence relation over $L_{\mathcal{G}}$.

PROOF The reflexivity and symmetry of equality follows from its definition. Transitivity: the definition of equality implies that

$$\begin{aligned} A = B &\Rightarrow \forall a \in A \bullet B \textit{ covers } a \\ &\Rightarrow \forall a \in A \bullet \exists b_1, b_2 \in B \bullet b_1 \leq a \wedge a \leq b_2 \end{aligned} \tag{20}$$

Similarly,

$B = C \Rightarrow \forall b \in B$ [illegible] $_1, c2 \in C \bullet c_1 \leq b \wedge b \leq c_2$ (21)

Combining (20) and (21) gives

$$\begin{aligned} A = B \wedge B = C \Rightarrow \forall a \in A \bullet \exists b_1, b_2 \in B, c_1, c_2 \in C \bullet \\ b_1 \leq a \wedge a \leq b_2 \wedge c_1 \leq b_1 \\ \wedge b_1 \leq c_2 \wedge c_3 \leq b_2 \wedge b_2 \leq c_4 \end{aligned}$$

Transitivity of bound order implies that

$$\begin{aligned} A = B \wedge B = C &\Rightarrow \forall a \in A \bullet \exists c_1, c_4 \in C \bullet c_1, c_4 \textit{ covers } a \\ &\Rightarrow \forall a \in A \bullet C \textit{ covers } a \end{aligned} \tag{22}$$

We can similarly show that

$$A = B \wedge B = C \Rightarrow \forall c \in C \bullet A \textit{ covers } c \tag{23}$$

Combining (22) with (23) gives,

$$A = B \wedge B = C \Rightarrow A = C$$

i.e., equality is transitive. □

Lemma 3 Equality preserves the group flow relation of $L_{\mathcal{G}}$, i.e.,

$$\forall A, B, X \in \alpha L_{\mathcal{G}} \bullet A \rightsquigarrow B \wedge A = X \Rightarrow X \rightsquigarrow B$$

PROOF The definition of equality gives

$$A = X \Rightarrow \forall a \in A \bullet \exists x \in X \bullet a \leq x \tag{24}$$

The definition of the group flow relation gives, for $A, B \in \alpha L_{\mathcal{G}}$,

$$A \rightsquigarrow B \Rightarrow \exists a \in A, b \in B \bullet a \rightsquigarrow b \tag{25}$$

However, equation (24) implies that for $a \in A$ in equation (25), there exists an $x \in X$ such that $x \leq b$, thus

$$\begin{aligned} A = X \wedge A \rightsquigarrow B &\Rightarrow \exists x \in X, b \in B \bullet x \leq b \\ &\Rightarrow X \rightsquigarrow B \end{aligned}$$

We can similarly show that

$$A \rightsquigarrow B \wedge B = X \Rightarrow A \rightsquigarrow X$$

□

Lemma 4 Equality preserves upper aggregate, i.e.,

$$\forall X, Y, A \in \alpha L_{\mathcal{G}} \bullet X = Y \Rightarrow A \oplus X = A \oplus Y$$

PROOF The definition of equality gives, for $X, Y \in \alpha L_{\mathcal{G}}$,

$$X = Y \Rightarrow \forall x \in X \bullet \exists y \in Y \bullet x \leq y$$

Since L forms a lattice, then for any $a \in \alpha R$

$$x \leq y \Rightarrow x \vee a \leq y \vee a$$

Thus for an arbitrary $A \in \alpha L_{\mathcal{G}}$,

$$X = Y \Rightarrow \forall a \in A, x \in X \bullet \exists y \in Y \bullet x \vee a \ \leq \ y \vee a \tag{26}$$

But given that $A \oplus B = \{a \vee b | a \in A, b \in B\}$, equation (26) can be written as

$$X = Y \Rightarrow \forall x \in A \oplus X \bullet \exists y \in A \oplus Y \bullet x \ \leq \ y \tag{27}$$

We can similarly show that

$$\begin{aligned} X = Y \quad &\Rightarrow \quad \forall x \in X \bullet \exists y \in Y \bullet y \ \leq \ x \\ &\Rightarrow \quad \forall x \in A \oplus B \bullet \exists y \in A \oplus Y \bullet y \ \leq \ x \end{aligned} \tag{28}$$

Equations (27) and (28) give

$$X = Y \Rightarrow \forall x \in A \oplus X \bullet A \oplus Y \ \mathit{covers} \ x$$

and we can similarly derive from $X = Y$ that,

$$X = Y \Rightarrow \forall y \in A \oplus Y \bullet A \oplus X \ \mathit{covers} \ y$$

and thus $X = Y$ implies that $A \oplus X = A \oplus Y$ □

Note that the proof that equality preserves the lower aggregate is identical in approach to lemma 4, and is not given here.

Lemma 5 Union of confinement groups is closed over equivalence classes, i.e. for $X, Y \in \alpha L_{\mathcal{G}}$

$$X = Y \Rightarrow X \cup Y = Y$$

PROOF From the definition of equality we have

$$\begin{aligned} X = Y \quad &\Rightarrow \quad \forall x \in X \bullet Y \ \mathit{covers} \ x \\ Y = Y \quad &\Rightarrow \quad \forall y \in Y \bullet Y \ \mathit{covers} \ y \end{aligned}$$

combining these give,

$$\forall x \in X \cup Y \bullet Y \ \mathit{covers} \ x \tag{29}$$

If X *covers* y ($X \in \alpha L_{\mathcal{G}}, y \in \alpha L$) then for any $A \in \alpha L_{\mathcal{G}}$, $X \cup A$ *covers* y also holds. Thus

$$X = Y \Rightarrow \forall y \in Y \bullet X \cup Y \ \mathit{covers} \ y \tag{30}$$

and combining (29) and (30) gives $X \cup Y = Y$ □

Lemma 6 Intersection of confinement groups is closed over equivalence classes, i.e. for $X, Y \in \alpha L_{\mathcal{G}}$

$$X = Y \Rightarrow X \cap Y = Y$$

PROOF We have,

$$X = Y \quad \Rightarrow \quad \forall y \in Y \bullet X \text{ } covers \text{ } y \tag{31}$$

$$X = Y \quad \Rightarrow \quad \forall x \in X \bullet Y \text{ } covers \text{ } x \tag{32}$$

Combining equation (31) with the fact that

$$Y = Y \Rightarrow \forall y \in Y \bullet Y \text{ } covers \text{ } y$$

gives

$$X = Y \Rightarrow \forall y \in X \cap Y \bullet Y \text{ } covers \text{ } y \tag{33}$$

Now, for any $y \in Y$, equation (32) gives

$$X = Y \Rightarrow \exists x_0 \in X \bullet y \leq x_0$$

Applying equation (31) to X_0 defined above, gives

$$X = Y \Rightarrow \exists x_0 \in X, y_0 \in Y \bullet y \leq x_0 \wedge x_0 \leq y_0$$

Repeatedly applying equations (31) and (32) gives and equation of the form ($n \geq 0$)

$$\begin{aligned} X = Y \quad \Rightarrow \quad & \exists x_0, \ldots, x_n \in X, y_0, \ldots, y_n \in Y \\ & \bullet y \leq x_0 \wedge x_0 \leq y_0 \wedge y_0 \leq x_1 \wedge \ldots \wedge x_n \leq y_n \end{aligned} \tag{34}$$

If $n < |Y|$, then there exists in (34) above, components y_h, y_k ($h < k$) such that $y_h = y_k$. Equation (34) above, implies that there exists an $x \in X$ such that $y_h \leq x \leq y_k$. Since $\leq$ is a partial order we have $y_h = x$. Therefore we can state that

$$\begin{aligned} X = Y \quad & \Rightarrow \quad \forall y \in Y \bullet \exists x \in X, y' \in Y \bullet y \leq x \wedge x = y' \\ & \Rightarrow \quad \forall y \in Y \bullet \exists x \in X \cap Y \bullet y \leq x \end{aligned} \tag{35}$$

We can similarly show that

$$\begin{aligned} X = Y \quad \Rightarrow \quad & \exists x_0, \ldots, x_n \in X, y_0, \ldots, y_n \in Y \\ & \bullet x_0 \leq y \wedge y_0 \leq x_0 \wedge x_1 \leq y_0 \wedge \ldots \wedge y_n \leq x_n \\ \Rightarrow \quad & \forall y \in Y \bullet \exists x \in X \cap Y \bullet x \leq y \end{aligned} \tag{36}$$

Equations (35) and (36) gives

$$X = Y \Rightarrow \forall y \in Y \bullet X \cap Y \; covers \; y$$

which when combined with (33) gives $X = Y \Rightarrow X \cap Y = Y$. □

Lemma 7 Equality preserves the flow relation $\rightsquigarrow$ defined on confinement groups, i.e. for $A, B, X \in \alpha L_{\mathcal{G}}$,

$$A \rightsquigarrow B \wedge A = X \Rightarrow X \rightsquigarrow B$$

PROOF We have by definition

$$\begin{aligned} A = X &\Rightarrow \forall a \in A \bullet \exists x \in X \bullet x \leq a \\ A \rightsquigarrow B &\Rightarrow \exists a \in A, b \in B \bullet a \leq b \end{aligned}$$

by transitivity, we have $x \leq a \wedge a \leq b \Rightarrow x \leq b$, thus

$$\begin{aligned} A = X \wedge A \rightsquigarrow B &\Rightarrow \exists x \in X shtx \leq b \\ &\Rightarrow X \rightsquigarrow B \end{aligned}$$

□

Lemma 8 The relation $\leq$ forms a partial order over the group policy $L_{\mathcal{G}}$.
PROOF Reflexivity follows from the definition of the relation.
Antisymmetry: Given $A, B \in \alpha L_{\mathcal{G}}$, then we have

$$\begin{aligned} A \leq B \wedge B \leq A &\Rightarrow \forall b \in B \bullet \exists a_1, a_2 \in A \bullet a_1 \leq b \wedge b \leq a_2 \wedge \\ &\quad\; \forall a \in A \bullet \exists b_1, b_2 \in B \bullet b_1 \leq a \wedge a \leq b_2 \\ &\Rightarrow \forall b \in B \bullet A \; covers \; b \wedge \\ &\quad\; \forall a \in A \bullet B \; covers \; a \\ &\Rightarrow A = B \end{aligned}$$

Transitivity: Given $A, B, C \in \alpha L_{\mathcal{G}}$, we have

$$\begin{aligned} A \leq B \wedge B \leq C \Rightarrow \; & \forall b \in B \bullet \exists a \in A \bullet a \leq b \wedge && (37) \\ & \forall a \in A \bullet \exists b \in B \bullet a \leq b \wedge && (38) \\ & \forall c \in C \bullet \exists b \in B \bullet b \leq c \wedge && (39) \\ & \forall b \in B \bullet \exists c \in C \bullet b \leq c && (40) \end{aligned}$$

Combining (37) with (39) and (38) with (40), transitivity gives

$$\begin{aligned} A \leq B \wedge B \leq C &\Rightarrow \forall c \in C \bullet \exists a \in A \bullet a \leq c \wedge \\ &\quad\ \forall a \in A \bullet \exists c \in C \bullet a \leq c \\ &\Rightarrow A \leq C \end{aligned}$$

□

Lemma 9 The equality relation preserves ordering on confinement groups, i.e. for $A, B, X \in \alpha L_{\mathcal{G}}$,

$$A \leq B \wedge A = X \Rightarrow X \leq B$$

PROOF We have by definition

$$\begin{aligned} A = X &\Rightarrow \forall a \in A \bullet \exists x \in X \bullet x \leq a \\ A \leq B &\Rightarrow \forall b \in B \bullet \exists a \in A \bullet a \leq b \end{aligned}$$

Combining these and noting the transitivity of bound order gives,

$$A = X \wedge A \leq B \Rightarrow \forall b \in B \bullet \exists x \in X \bullet x \leq B \tag{41}$$

Similarly, we have

$$\begin{aligned} A = X &\Rightarrow \forall z \in X \bullet \exists a \in A \bullet x \leq a \\ A \leq B &\Rightarrow \forall a \in A \bullet \exists b \in B \bullet a \leq b \end{aligned}$$

which implies

$$A = X \wedge A \leq B \Rightarrow \forall x \in X \bullet \exists b \in B \bullet x \leq b \tag{42}$$

Combining (41) and (42) gives

$$A = X \wedge A \leq B \Rightarrow X \leq B$$

□

Lemma 10 Given partial ordering $\leq$ on $L_{\mathcal{G}}$, the upper aggregate operator gives an upper bound on its operands, i.e. for $A, B \in \alpha L_{\mathcal{G}}$ then $A \leq A \oplus B$ and $B \leq A \oplus B$

PROOF Reflexivity implies that

$$\begin{aligned} & A \oplus B \le A \oplus B && (43)\\ \Rightarrow\ & \forall x \in A \oplus B \bullet \exists y \in A \oplus B \bullet x \le y && \\ \Rightarrow\ & \forall a \in A, b \in B \bullet \exists y \in A \oplus B \bullet a \vee b \le y && \\ \Rightarrow\ & \forall a \in A \bullet \exists y \in A \oplus B \bullet a \vee b \le y && (44) \end{aligned}$$

(since $a \vee b \le y$ implies $a \le y$). Similarly,

$$\begin{aligned} & A \oplus B \le A \oplus B && (45)\\ \Rightarrow\ & \forall y \in A \oplus B \bullet \exists a \in A, b \in B \bullet a \vee b \le y && \\ \Rightarrow\ & \forall y \in A \oplus B \bullet \exists a \in A \bullet a \le y && (46) \end{aligned}$$

Combining (44) and (46) gives $A \le A \oplus B$. We can similarly show that $B \le A \oplus B$ also holds. □

The proof that the lower aggregate operator gives a lower bound on its operands (under the ordering relation $\le$) is identical in approach to lemma 10 above, and is not given here.

Lemma 11 The aggregation operators $\oplus$ and $\otimes$ are associative.
PROOF Follows, since the upper and lower bounds operators defined on the base lattice L are associative, i.e., for $A, B \in \alpha L_{\mathcal{G}}$,

$$\begin{aligned} (A \oplus B) \oplus C &= \{x \vee c | (x \in A \oplus B) \wedge c \in C\} \\ &= \{(a \vee b) \vee c | a \in A \wedge b \in B \wedge c \in C\} \\ &= \{a \vee (b \vee c) | a \in A \wedge b \in B \wedge c \in C\} \\ &= \{a \vee x | a \in A \wedge x \in (B \vee C)\} \\ &= A \oplus (B \oplus C) \end{aligned}$$

and similarly for the lower aggregate operator. □

Lemma 12 If a confinement group policy is constructed from a distributive lattice then the aggregate operators are distributive, i.e., for $A, B, C \in L_{\mathcal{G}}$, then

$$A \oplus (B \otimes C) = (A \oplus B) \otimes (A \otimes C)$$

PROOF We have by definition and the distributivity of L,

$$\begin{aligned} A \oplus (B \otimes C) &= A \oplus \{b \wedge c | b \in B \wedge c \in C\} \\ &= \{a \vee (b \wedge c) | a \in A \wedge b \in B \wedge c \in C\} \\ &= \{(a \vee b) \wedge (a \vee c | a \in A \wedge b \in B \wedge c \in C\} \\ &= \{x \wedge y | x \in (A \oplus B) \wedge y \in (A \oplus C)\} \\ &= (A \oplus B) \otimes (A \otimes C) \end{aligned}$$

It follows that

$$A \otimes (B \oplus C) = (A \otimes B) \oplus (A \otimes C)$$

also holds. Note that if our base lattice L is not distributive then $L_{\mathcal{G}}$ is not distributive. □

Lemma 13 Given confinement groups $A, B \in \alpha L_{\mathcal{G}}$ then,

$$\forall D \bullet (A \leq D \wedge B \leq D) \Rightarrow A \oplus B \leadsto D$$

PROOF We have by definition of $\leq$

$$\begin{aligned} A \leq D \wedge B \leq D &\Rightarrow \forall d \in D \bullet \exists a \in A, b \in B \bullet a \leq d \wedge b \leq d \\ &\Rightarrow \forall d \in D \bullet \exists a \in A, b \in B \bullet a \vee b \leq d \\ &\Rightarrow \forall d \in D \bullet \exists x \in A \oplus B \bullet x \leq d \\ &\Rightarrow A \oplus B \leadsto D \end{aligned}$$

□

Lemma 14 Given confinement groups $A, B \in \alpha L_{\mathcal{G}}$ then,

$$\forall D \bullet (A \leq D \wedge B \leq D) \Rightarrow \exists X \subseteq A \oplus B \bullet A \leq X \wedge B \leq X \wedge X \leq D$$

PROOF Define X above as

$$X = \{x \in A \oplus B | \exists d \in D \bullet x \leq d\}$$

It follows from its definition that

$$\forall x \in X \bullet \exists d \in D \bullet x \leq d \tag{47}$$

Since D is an upper bound on A and B then,

$$\begin{aligned} & \forall d \in D \bullet \exists a \in A, b \in B \bullet a \vee b \ \leq \ b \\ \Rightarrow \ & \forall d \in D \bullet \exists x \in A \oplus B \bullet x \ \leq \ d \end{aligned}$$

Thus for any $d \in D$, we have an $x \in A \oplus B$ with $x \ \leq \ d$, and therefore x is also a member of X defined above. Thus,

$$\forall d \in D \exists x \in X \bullet x \ \leq \ d \tag{48}$$

Combining equations (47) and (48) gives $X \ \leq \ D$. Next, from the definition of X we have,

$$\begin{aligned} & \forall x \in X \bullet \exists a \in A, b \in B \bullet a \vee b = x \qquad (49) \\ \Rightarrow \ & \forall x \in X \bullet \exists a \in A \bullet a \ \leq \ x \end{aligned}$$

Since $A \ \leq \ D$ then,

$$\forall a \in A \bullet \exists d \in D \bullet a \ \leq \ d \tag{50}$$

and since $B \ \leq \ D$ we have

$$\forall d \in D \bullet \exists b \in B \bullet b \ \leq \ d \tag{51}$$

combining (50) and (51) gives

$$\begin{aligned} & \forall a \in A \bullet \exists d \in D, b \in B \bullet a \ \leq \ d \wedge b \ \leq \ d \\ \Rightarrow \ & \forall a \in A \bullet \exists b \in B \bullet \exists d \in D \bullet a \ \leq \ a \vee b \wedge a \vee b \ \leq \ d \end{aligned}$$

and since X contains $a \vee b$ ($a \in A, b \in B$) such that $a \vee b \ \leq \ d$ (for some $d \in D$), we can write

$$\forall a \in A \bullet \exists x \in X \bullet a \ \leq \ X$$

combining this with (50) above gives $A \ \leq \ X$, and we similarly have $B \ \leq \ X$, and the lemma is proven. □

A.2 State Machine Model SMM

Lemma 15 Given confinement groups A, B and C, each constrained so that $\nabla(A), \nabla(B)$ and $\nabla(C)$ holds, then

$$C \leadsto A \otimes B \Leftrightarrow C \leadsto A \wedge C \leadsto B$$

PROOF I We have

$$\begin{aligned} C \leadsto A \otimes B &\Rightarrow \exists c \in C, a \in A, b \in B \bullet c \leq a \wedge b \\ &\Rightarrow \exists c \in C, a \in A, b \in B \bullet c \leq a \wedge c \leq b \\ &\Rightarrow C \leadsto A \wedge C \leadsto B \end{aligned}$$

PROOF II We have

$$C \leadsto A \wedge C \leadsto B \Rightarrow \exists c, c' \in C, a \in A, b \in B \bullet c \leq a \wedge c' \leq b$$

but because $\nabla(C)$ holds, we have $\bot_C \leq a$ and $\bot_C \leq b$. Thus

$$\begin{aligned} C \leadsto A \wedge C \leadsto B &\Rightarrow \exists c \in C, a \in A, b \in B \bullet c \leq a \vee b \\ &\Rightarrow C \leadsto A \otimes B \end{aligned}$$

Note that

$$A \oplus B \leadsto C \Leftrightarrow A \leadsto C \wedge B \leadsto C$$

does not necessarily hold (in fact, for this to hold, C must contain a component that forms an upper bound on every element of C). □

Lemma 16 $\nabla()$ is closed over $\oplus$ and $\otimes$, i.e., for groups A and B,

$$\begin{aligned} \nabla(A) \wedge \nabla(B) &\Rightarrow \nabla(A \oplus B) \\ &\Rightarrow \nabla(A \otimes B) \end{aligned}$$

PROOF

$$\begin{aligned} \nabla(A) \wedge \nabla(B) &\Rightarrow \bot_A \vee \bot_B \in A \oplus B \\ &\Rightarrow \nabla(A \oplus B) \\ &\Rightarrow \bot_A \wedge \bot_B \in A \otimes B \\ &\Rightarrow \nabla(A \otimes B) \end{aligned}$$

□

Lemma 17 Given confinement groups A and B with $\nabla(A) \wedge \nabla(B)$, then

$$\bot_{A \oplus B} \Leftrightarrow \bot_A \vee \bot_B$$

PROOF Follows from the definition $A \oplus B = \{a \vee B | a \in A \wedge b \in B\}$ □

Lemma 18 Given entities E and F, and a secure transition sequence Σ_n, then

$$E \overset{\Sigma_n}{\rhd} F \Rightarrow \forall f \in s_n.\underline{F} \bullet \perp_{s_0.\underline{E}} \leq f$$

PROOF We will use induction on the length of Σ_n.

- If $n = 0$, then since s_0 is secure we have,

  $$\begin{aligned} E \overset{\Sigma_n}{\rhd} F &\Leftrightarrow E \overset{s_0}{\rhd} F \\ &\Rightarrow \forall f \in s_n.\underline{F} \bullet \perp_{s_0.\underline{E}} \leq f \end{aligned}$$

  and thus the trivial case holds.

- (inductive step) Assume that it holds for n. Now consider Σ_{n+1}. We have by definition,

  $$E \overset{\Sigma_{n+1}}{\rhd} F \Rightarrow \exists G \in ENTS \bullet E \overset{\Sigma_n}{\rhd} G \wedge G \overset{s_{n+1}}{\rhd} F$$

  The inductive hypothesis gives,

  $$E \overset{\Sigma_n}{\rhd} G \Rightarrow \forall g \in s_n.\underline{G} \bullet \perp_{s_0.\underline{E}} \leq g \tag{52}$$

  The definition of a secure state gives,

  $$G \overset{s_{n+1}}{\rhd} F \Rightarrow \forall f \in s_{n+1}.\underline{F} \bullet \perp_{s_{n+1}.\underline{G}} \leq f \tag{53}$$

  Since the transition from s_n to s_{n+1} is secure we have $s_{n+1}.\underline{G} \subseteq s_n.\underline{G}$. Applying this to equations (52) and (53) gives

  $$\begin{aligned} E \overset{\Sigma_{n+1}}{\rhd} F &\Rightarrow \forall f \in s_{n+1}.\underline{F} \bullet \perp_{s_n.\underline{G}} \leq f \wedge \perp_{s_0.\underline{E}} \leq \perp_{s_n.\underline{G}} \\ &\Rightarrow \forall f \in s_{n+1}.\underline{F} \bullet \perp_{s_0.E} \leq f \end{aligned}$$

Thus, by induction the lemma holds. □

Theorem 2 The state model SMM is a refinement of the abstract GCFM, i.e., for every secure transition sequence Σ_n, then

$$\forall \mathcal{E}, \mathcal{F} \subseteq ENTS \bullet \mathcal{E} \overset{\Sigma_n}{\rhd} \mathcal{F} \Rightarrow s_0.\underline{\mathcal{E}}_\oplus \overset{\bullet}{\leadsto}_0 .\underline{\mathcal{F}}_\otimes$$

If this holds, then any system that is secure by SMM (every Σ_n is secure) will be secure by GCFM.

PROOF We have for $\mathcal{E}, \mathcal{F} \subseteq ENTS$

$$\mathcal{E} \overset{\Sigma_n}{\rhd} \mathcal{F} \Rightarrow \forall E \in \mathcal{E}, F \in \mathcal{F} \bullet E \overset{\Sigma_n}{\rhd} F$$

Applying the previous lemma (18) gives,

$$\begin{aligned} &\forall E \in \mathcal{E}, F \in \mathcal{F} \bullet \forall f \in s_n.\underline{F} \bullet \perp_{s_0.\underline{E}} \leq f \\ \Rightarrow\ &\forall F \in \mathcal{F} \bullet \forall f \in s_n.\underline{F} \bullet \perp_{s_0.\underline{\mathcal{E}}_{\oplus}} \leq f \end{aligned}$$

Since the transitions from state s_0 to state s_n are secure then $s_n.\underline{F} \subseteq s_0.\underline{F}$, and thus

$$\begin{aligned} \mathcal{E} \overset{\Sigma_n}{\rhd} \mathcal{F} \ &\Rightarrow\ \forall F \in \mathcal{F} \bullet \exists f \in s_0.\underline{F} \bullet \perp_{s_0.\underline{\mathcal{E}}_{\oplus}} \leq f \\ &\Rightarrow\ \forall F \in \mathcal{F} \bullet s_0.\underline{\mathcal{E}}_{\oplus} \rightsquigarrow \underline{F} \end{aligned}$$

since the confinement groups are restricted, lemma 15 can be called on to give

$$\mathcal{E} \overset{\Sigma_n}{\rhd} \mathcal{F} \Rightarrow s_0.\underline{\mathcal{E}}_{\oplus} \rightsquigarrow s_0.\underline{\mathcal{F}}_{\otimes}$$

Lemma 19 A (secure) transition from a secure state s to a secure state s' with flows described by $\overset{\bullet}{\rhd}$ is possible iff

$$\forall E \in ENTS \bullet \oplus \{s.\underline{X} | X \overset{\bullet}{\rhd} E\} \rightsquigarrow s.\underline{E} \tag{54}$$

PROOF I (If a secure transition to secure s' with flows $\overset{\bullet}{\rhd}$ at state s' exists then equation 54 holds) At state s' we have for $E \in ENTS$,

$$\begin{aligned} \forall X \in ENTS \bullet X \overset{s'}{\rhd} E \ &\Rightarrow\ \forall e \in s'.\underline{E} \bullet \perp_{s'.\underline{X}} \leq e \\ &\Rightarrow\ \oplus \{s'.\underline{X} | X \overset{\bullet}{\rhd} E\} \rightsquigarrow s'.\underline{E} \end{aligned}$$

since the transition from s to s' is secure we have $s'.\underline{X} \subseteq s.\underline{X}$ for each entity X and thus

$$\oplus \{s'.\underline{X} | X \overset{\bullet}{\rhd} E\} \subseteq \oplus \{s.\underline{X} | X \overset{\bullet}{\rhd} E\}$$

which implies,

$$\forall E \in ENTS \bullet \oplus \{s.\underline{X} | X \overset{\bullet}{\rhd} E\} \rightsquigarrow s.\underline{E}$$

PROOF II (if equation (54) holds, then there exists a secure state s' with flows described by relation $\overset{\bullet}{\rhd}$, and reached by a secure transition from s) Define the confinement of each entity E at state s' as

$$s'.\underline{E} \doteq \{s.\underline{X} | X \overset{\bullet}{\rhd} E\} \cap s.\underline{E}$$

and the flows at state s' as: $E \overset{s'}{\rhd} F \Leftrightarrow E \overset{\bullet}{\rhd} F$.

- Before proving that s' and the transition is secure we will prove that the confinement for each entity is non-empty. Consider the lowest bound on $A = \oplus \{s.\underline{X} | X \overset{*}{\triangleright} E\}$. Since (54) holds, we know that there exists some $e \in s.\underline{E}$ with $\perp_A \leq e$. Futhermore, since $\overset{*}{\triangleright}$ is reflexive we have $E \overset{*}{\triangleright} E$ and thus $\perp_{s.\underline{E}} \leq \perp_A$ for A above. Thus we have

$$\perp_{s.\underline{E}} \leq \perp_A \leq e \tag{55}$$

for $\perp_{s.\underline{E}}, e \in s.\underline{E}$. This implies that $s.\underline{E}$ *covers* $\perp_A$. Thus we have $\perp_A \in s.\underline{E}$, and also $\perp_A \in A$, and therefore $\perp_A \in s'.\underline{E}$.

- That the transition from s to s' is secure follows since by its defintion $s'.\underline{E}$ is a subset of $s.\underline{E}$.

- (state s' is secure) Firstly, the group confinement of each entity is valid since we know that it is non-empty, and by lemma 15 since $\nabla(s.\underline{X})$ holds then $\nabla(\oplus \{s.\underline{X} | X \triangleright E\})$ also holds.

  Now, suppose that there is a flow $E \overset{*}{\triangleright} F$ at state s'. We know by the transitivity of $\overset{*}{\triangleright}$ that,

$$\begin{aligned} & s'.\underline{F} = (\oplus \{s.\underline{X} | X \overset{*}{\triangleright} E\} \oplus \oplus \{s.\underline{Y} | Y \overset{*}{\triangleright} B \wedge \neg Y \overset{*}{\triangleright} A\}) \cap s.\underline{F} \\ & \Rightarrow \forall f \in s'.\underline{F} \bullet \perp_{\oplus\{s.\underline{X}|X \overset{*}{\triangleright} E\}} \leq f \end{aligned}$$

  From above (equation 55) we know that this lower bound is also a member of $s'.\underline{E}$, and since

$$s'.\underline{E} = \oplus \{s.\underline{X} | X \overset{*}{\triangleright} E\} \cap s.\underline{E}$$

  we have

$$E \overset{*}{\triangleright} F \Rightarrow \forall f \in s'.\underline{F} \bullet \perp_{s'.\underline{E}} \leq f$$

  and state s' is secure.

□

Lemma 20 Given confinement groups A and B such that $\nabla(A) \wedge \nabla(B)$, then

$$(A \oplus B) \cap A = \{a \in A | \perp_B \leq a\}$$

PROOF We have by defintion of upper aggregate,

$$a \in (A \oplus B) \cap A \quad \Rightarrow \quad a \in A \wedge \exists a' \in A, b \in B \bullet a' \overset{b}{\vee} = a$$

$$\begin{array}{rcl} & \Rightarrow & a \in A \wedge \exists b \in B \bullet b \leq a \\ & \Rightarrow & a \in A \wedge \perp_B \leq a \\ a \in A \wedge \perp_B \leq & \Rightarrow & \exists b \in B \bullet a \in A \wedge b \leq a \\ & \Rightarrow & \exists b \in B \bullet a \in A \wedge a \vee b = a \\ & \Rightarrow & a \in (A \oplus B) \cap A \end{array}$$

□

Lemma 21 Given A and B such that $\nabla(A) \wedge \nabla(B)$, then

$$\begin{array}{rcl} B \leadsto A & \Rightarrow & \nabla((A \oplus B) \cap A) \\ & \Rightarrow & \perp_{(A \oplus B) \cap A} = \perp_A \vee \perp_B \end{array}$$

PROOF We will prove that $\perp_A \vee \perp_B$ is a member of $(A \oplus B) \cap A$, and since it forms a lowest bound on $(A \oplus B)$ it will also form a lowest bound on $(A \oplus B) \cap A$ and thus $\nabla((A \oplus B) \cap A)$ holds.

Firstly, note that $\perp_A \vee \perp_B$ is a member of $A \oplus B$. We have,

$$B \leadsto A \Rightarrow \exists a \in A, b \in B \bullet b \leq a$$

but $\nabla(B)$ holds, so this simplifies to

$$\begin{array}{rcl} B \leadsto A & \Rightarrow & \exists a \in A \bullet \perp_B \leq a \\ & \Rightarrow & \exists a \in A \bullet \perp_B \vee \perp_A \leq a \vee \perp_A \\ & \Rightarrow & \exists a \in A \bullet \perp_B \vee \perp_A \leq a \end{array}$$

This and the fact that $\perp_A \leq \perp_B \vee \perp_A$ trivially holds gives

$$\begin{array}{rcl} B \leadsto A & \Rightarrow & \exists a \in A \bullet \perp_A \leq \perp_B \vee \perp_A \wedge \perp_B \vee \perp_A \leq a \\ & \Rightarrow & A \ covers \ \perp_A \vee \perp_B \\ & \Rightarrow & \perp_A \vee \perp_B \in A \end{array}$$

Thus we have $\perp_A \vee \perp_B \in (A \oplus B) \cap A$. □

Lemma 22 The confinement of an entity S at state s_{n+1} reached by transition sequence Σ^R_{n+1} (i.e, transition sequence Σ_{n+1} built up using transition Req), can be calculated as,

$$s_{n+1}.\underline{E} = \{e \in s_n.\underline{E} | \forall X \bullet X \overset{s_{n+1}}{\rhd} E \Rightarrow \perp_{s_n.\underline{X}} \leq e\}$$

PROOF Since $\triangleright$ is reflexive, we have $E \overset{s_{n+1}}{\triangleright} E$ and applying lemma 20 gives,

$$\begin{aligned} s_{n+1}.\underline{E} &= \oplus\{s_n.\underline{X}|X \overset{s_{n+1}}{\triangleright} E\} \cap s_n.\underline{E} \\ &= (\oplus\{s_n.\underline{X}|X \overset{s_{n+1}}{\triangleright} E\} \oplus s_n.\underline{E}) \cap s_n.\underline{E} \\ &= \{e \in s_n.\underline{E}| \perp_{\oplus\{s_n.\underline{X}|X \overset{s_{n+1}}{\triangleright} E\}} \leq e\} \end{aligned}$$

Finally, applying lemma 17 to this gives

$$s_{n+1}.\underline{E} = \{e \in s_n.\underline{E}|\forall X \bullet X \overset{s_{n+1}}{\triangleright} E \Rightarrow \perp_{s_n.\underline{X}} \leq e\}$$

the required result. □

Lemma 23 The confinement of an entity E at a state s_n reached by transition sequence Σ_n^R can be calculated as,

$$s_n.\underline{E} = \{e \in s_0.\underline{E}|X \overset{\Sigma_n^R}{\triangleright} E\} \cap s_0.\underline{E} \tag{56}$$

PROOF First note that the application of lemmas 20 and 17 can produce a result similar to lemma 22 by allowing us to write the equation (56) as

$$s_n.\underline{E} = \{e \in s_0.\underline{E}|\forall X \bullet X \overset{\Sigma_n^R}{\triangleright} E \Rightarrow \perp_{s_0.\underline{X}} \leq e\} \tag{57}$$

To prove that this equation (57) holds (and thus equation (56) holds), we will use induction on the length of Σ_n^R.

- If $n = 1$, then

  $$s_1.\underline{E} = \oplus\{s_0.\underline{X}|X \overset{s_1}{\triangleright} E\} \cap s_0.\underline{E}$$

  follows from the post-condition for transition function *Req*.

- Assume that (57) holds for n (hypothesis). Now consider a transition from state s_n to state s_{n+1}. Since the transition is made by *Req*, then

  $$s_{n+1}.\underline{E} = \{e \in s_n.\underline{E}|\forall X \bullet X \overset{s_{n+1}}{\triangleright} E \Rightarrow \perp_{s_n.\underline{X}} \leq e\} \tag{58}$$

  Consider each X in this equation (58), where $X \overset{s_{n+1}}{\triangleright} E$. We have by the inductive hypothesis,

  $$s_{n+1}.\underline{X} = \{x \in s_0.\underline{X}|\forall Y \bullet Y \overset{\Sigma_n^R}{\triangleright} X \Rightarrow \perp_{s_0.\underline{Y}} \leq x\}$$

and since the original policy is a lattice,

$$(\bot_{s_n.\underline{X}} \leq e) \Leftrightarrow (\forall Y \bullet Y \overset{\Sigma_n^R}{\rhd} X \Rightarrow \bot_{s_0.\underline{Y}} \leq e)$$

Thus (58) can be written as,

$$\begin{aligned} s_{n+1}.\underline{E} &= \{e \in s_n.\underline{E} | \forall X \bullet X \overset{s_{n+1}}{\rhd} E \Rightarrow \forall Y \bullet Y \overset{\Sigma_n^R}{\rhd} X \Rightarrow \bot_{s_0.\underline{Y}} \leq e\} \\ &= \{e \in s_n.\underline{E} | \forall X, Y \bullet Y \overset{\Sigma_n^R}{\rhd} X \wedge X \overset{\Sigma_{n+1}}{\rhd} E \Rightarrow \bot_{s_0.\underline{Y}} \leq e\} \\ &= \{e \in s_n.\underline{E} | \forall Y \bullet Y \overset{\Sigma_{n+1}^R}{\rhd} E \Rightarrow \bot_{s_0.\underline{Y}} \leq e\} \qquad (59) \end{aligned}$$

Furthermore, if $e \in s_n.\underline{E}$, the inductive hypothesis implies,

$$e \in s_0.\underline{E} \wedge \forall X \bullet X \overset{\Sigma_n^R}{\rhd} E \Rightarrow \bot_{s_0.\underline{X}} \leq e$$

But this condition already occurs in equation (59), and thus we can write,

$$\begin{aligned} s_{n+1}.\underline{E} &= \{e \in s_0.\underline{E} | \forall X \bullet X \overset{\Sigma_n^R}{\rhd} E \Rightarrow \bot_{s_0.\underline{X}} \leq e\} \\ &= \{e \in s_0.\underline{E} | X \overset{\Sigma_n^R}{\rhd} E\} \cap s_0.\underline{E} \end{aligned}$$

and the lemma is proven. □

Lemma 24 The lowest bound on the confinement of an entity E at a state s_n reached by transition Σ_n^R can be calculated as

$$\bot_{s_n.\underline{E}} = \vee \{\bot_{s_0.\underline{X}} | X \overset{\Sigma_n^R}{\rhd} E\}$$

PROOF From lemma 23, we have

$$\bot_{s_n.\underline{E}} = \bot_{(\oplus\{s_0.\underline{X}|X \overset{\Sigma_n^R}{\rhd} E\} \oplus s_0.\underline{E}) \cap s_0.\underline{E}}$$

Since state s_n is secure, $\oplus \{s_0.\underline{X} | X \overset{\Sigma_n^R}{\rhd} E\} \rightsquigarrow s_0.\underline{E}$ (theorem 2), applying lemma 21 gives

$$\bot_{s_n.\underline{E}} = \bot_{\oplus\{s_0.\underline{X}|X \overset{\Sigma_n^R}{\rhd} E\}} \vee \bot_{s_0.\underline{E}}$$

reflexivity of $\rhd$ gives $E \rhd E$ which, along with lemma 17 implies

$$\begin{aligned} \bot_{s_n.\underline{E}} &= \bot_{\oplus\{s_0.\underline{X}|X \overset{\Sigma_n^R}{\rhd} E\}} \\ &= \vee \{\bot_{s_0.\underline{X}} | X \overset{\Sigma_n^R}{\rhd} E\} \end{aligned}$$

□

Theorem 3 Given a transition sequence Σ_n^R, built from transition function *Req*, then if the transition to a state s_{n+1} with flows defined by $\overset{op}{\rhd}$ is not possible by the pre-condition on $Req(op, s)$, then the flows described by Σ_{n+1} are not secure by GCFM, i.e., given state s_n reached by Σ_n^R,

$$\exists E \in ENTS \bullet \oplus \{s_n.\underline{X} | X \overset{s_{n+1}}{\rhd} E\} \not\rightsquigarrow s_n.\underline{E} \Rightarrow$$
$$\neg(\forall \mathcal{E}, \mathcal{F} \subseteq ENTS \bullet \mathcal{E} \overset{\Sigma_{n+1}}{\rhd} \mathcal{F} \Rightarrow s_0.\underline{\mathcal{E}}_\oplus \rightsquigarrow s_0.\underline{\mathcal{F}}_\otimes)$$

PROOF To prove this we must, given $\oplus \{s_n.\underline{X} | X \overset{s_{n+1}}{\rhd} E\} \not\rightsquigarrow s_n.\underline{E}$, show that some $\mathcal{E}, \mathcal{F} \subseteq ENTS$ exist such that $\mathcal{E} \overset{\Sigma_{n+1}}{\rhd} \mathcal{F}$ but $s_0.\underline{\mathcal{E}}_\oplus \not\rightsquigarrow s_0.\underline{\mathcal{F}}_\otimes$. We will pick this $\mathcal{E}$ and $\mathcal{F}$ to be $\{X | X \overset{\Sigma_{n+1}}{\rhd} E\}$ and $\{E\}$ respectively.

Suppose there is an E such that $G \not\rightsquigarrow s_n.\underline{E}$, where

$$G = \oplus \{s_n.\underline{X} | X \overset{s_{n+1}}{\rhd} E\}$$

Applying lemmas 17 and 24 in order gives

$$\begin{aligned} \bot_G &= \vee \{\bot_{s_n.\underline{X}} | X \overset{s_{n+1}}{\rhd} E\} \\ &= \vee \{\vee \{\bot_{s_n.\underline{Y}} | Y \overset{\Sigma_n^R}{\rhd} X\} | X \overset{s_{n+1}}{\rhd} E\} \\ &= \vee \{\bot_{s_n.\underline{Y}} | \exists X \bullet Y \overset{\Sigma_n^R}{\rhd} X \wedge X \overset{s_{n+1}}{\rhd} E\} \\ &= \vee \{\bot_{s_n.\underline{Y}} | Y \overset{\Sigma_{n+1}}{\rhd} E\} \qquad (60) \end{aligned}$$

Note that since $E \overset{s_{n+1}}{\rhd} E$, we have $\{X | X \overset{\Sigma_n^R}{\rhd} E\} \subseteq \{X \overset{\Sigma_{n+1}^R}{\rhd} E\}$, which implies

$$\bot_{s_n.\underline{E}} \subseteq \bot_G \qquad (61)$$

We have,

$$G \not\rightsquigarrow s_n.\underline{E} \Leftrightarrow \forall e \in s_n.\underline{E} \bullet \neg \bot_G \leq e$$

Applying lemma 23 to $s_n.\underline{E}$ in this equation gives

$$\begin{aligned} G \not\rightsquigarrow s_n.\underline{E} &\Rightarrow \forall e \in s_0.\underline{E} \bullet (\forall X \bullet X \overset{\Sigma_n^R}{\rhd} E \Rightarrow \bot_{s_0.\underline{X}} \leq e) \Rightarrow \neg \bot_G \leq e \\ &\Rightarrow \forall e \in s_0.\underline{E} \bullet \bot_{\oplus\{s_0.\underline{X} | X \overset{\Sigma_n^R}{\rhd} E\}} \leq e \Rightarrow \neg \bot_G \leq e \\ &\Rightarrow \forall e \in s_0.\underline{E} \bullet \neg(\oplus \bot_{\{s_0.\underline{X} | X \overset{\Sigma_n^R}{\rhd} E\}} \leq e \wedge \bot_G \leq e) \end{aligned}$$

However, we have $\bot_{\oplus\{s_0.\underline{X} | X \overset{\Sigma_n^R}{\rhd} E\}} = \bot_{s_n.\underline{E}}$, and from equation (61), $\bot_{s_n.\underline{E}} \leq \bot_G$. This gives

$$G \not\rightsquigarrow s_n.\underline{E} \Rightarrow \forall e \in s_0.\underline{E} \bullet \neg \bot_G \leq e$$

Applying equation (60) gives,

$$G \not\rightsquigarrow s_n.\underline{E} \Rightarrow \oplus \{s_0.\underline{Y} | Y \overset{\Sigma_{n+1}}{\rhd} E\} \not\rightsquigarrow s_0.\underline{E}$$

Thus the flow $\{Y | Y \overset{\Sigma_{n+1}}{\rhd} E\} \overset{\Sigma_{n+1}}{\rhd} \{E\}$ which is not permitted by the transition function *Req* is not secure in GCFM. □

A.3 Reflexive Flow Policies

Lemma 25 The bound order relation (definition 9) is partially ordered.

PROOF Reflexivity and antisymmetry follows directly from the fact that the flow relation $\rightsquigarrow$ is reflexive and pseduo-antisymmetric.

By the definition of bound order we have for $a, b, c \in \alpha R$

$$\begin{aligned} a \leq b \wedge b \leq c \;&\Rightarrow\; (\forall x \bullet x \rightsquigarrow a \Rightarrow x \rightsquigarrow b) \wedge (\forall x \bullet x \rightsquigarrow b \Rightarrow x \rightsquigarrow c) \\ &\Rightarrow \forall x \bullet x \rightsquigarrow a \Rightarrow x \rightsquigarrow c \end{aligned} \tag{62}$$

similarly,

$$a \leq b \wedge b \leq c \Rightarrow \forall x \bullet c \rightsquigarrow x \Rightarrow a \rightsquigarrow x \tag{63}$$

combining equations (62) and (63) gives

$$a \leq b \wedge b \leq c \Rightarrow a \leq c$$

i.e., bound order is transitive. □

Theorem 4 The symmetric powerset lattice forms a lattice, i.e., given set S then $\mathcal{P}_S\, S$ is partially ordered and every pair of components have unique lowest upper and greatest lower bounds.

PROOF

- $\mathcal{P}_S\, S$ is reflexive. Follows from its definition.

- $\mathcal{P}_S\, S$ is antisymmetric. For $A, B \in \mathcal{P}_S\, S$, we have by definition,

  $$\begin{aligned} A \;\leq\; B &\;\Rightarrow\; B_L \subseteq A_L \\ B \;\leq\; A &\;\Rightarrow\; A_L \subseteq B_L \end{aligned}$$

  which implies $A_L = B_L$. We can similarly show that $A_H = B_H$, and thus $A = B$.

- $\mathcal{P}_S\, S$ is transitive. For $A, B, C \in \mathcal{P}_S\, S$, we have by definition,

  $$\begin{aligned} A \;\leq\; B &\;\Rightarrow\; B_L \subseteq A_L \\ B \;\leq\; C &\;\Rightarrow\; C_L \subseteq B_L \end{aligned}$$

  which implies $C_L \subseteq A_L$. We can similarly show that $C_H \supseteq A_H$, and thus $A \;\leq\; C$.

- $\vee$ is the lowest upper bound operator for $\mathcal{P}_S\, S$. Since intersection gives greatest lower bound and union lowest upper bound on a powerset lattice, then for $A, B \in \mathcal{P}_S\, S$,

  $$(A_L \supseteq A_L \cap B_L) \wedge (A_H \subseteq A_H \cup B_H)$$

  i.e., $A \;\leq\; A \vee B$. Similarly, $B \;\leq\; A \vee B$. Furthermore, for all $C \in \mathcal{P}_S\, S$ we have,

  $$(A_L \supseteq C_L \wedge B_L \supseteq C_L) \wedge (A_H \subseteq C_H \wedge B_H \subseteq C_H)$$

  which implies

  $$(A_L \cap B_L \supseteq C_L) \wedge (A_H \cup B_H \subseteq C_H)$$

  i.e, $A \vee B$ gives the lowest upper bound.

- $\wedge$ is the greatest lower bound operator. Follows from the symmetry between its definiton and the definition of the lowest upper bound operator.

□

Lemma 26 The Symmetric powerset lattice is distributive, i.e, for $A, B, C \in \mathcal{P}_S\, S$ for some set S, then

$$A \vee (B \wedge C) = (A \vee B) \wedge (A \wedge C)$$

PROOF Since a powerset lattice is distributive, then from the defintion of a symmetric powerset lattice,

$$\begin{aligned} A \vee (B \wedge C) &= A \vee (B_L \cup C_L, A_H \cap C_H) \\ &= (A_L \cap (B_L \cup C_L), A_H \cup (B_H \cap C_H)) \\ &= ((A_L \cap B_L) \cup (A_L \cap C_L), (A_H \cup B_H) \cap (A_H \cup C_H)) \\ &= (A \vee B) \wedge (A \wedge C) \end{aligned}$$

and we can simlarly show that

$$A \wedge (B \vee C) = (A \wedge B) \vee (A \wedge C)$$

also holds. □

Theorem 5 The mapping $f^{\mathcal{P}_S}$ from an arbitrary quoset Q to symmetric powerset lattice $\mathcal{P}_S\,\alpha Q$ preserves the orderings of Q, i.e.,

$$\forall a, b \in \alpha Q \bullet a \overset{Q}{\leq} b \Leftrightarrow f^{\mathcal{P}_S}(a) \overset{\mathcal{P}_S\alpha Q}{\leq} f^{\mathcal{P}_S}(b)$$

PROOF I For $a, b \in \alpha Q$ we have,

$$\begin{aligned} f^{\mathcal{P}_S}(a) \leq f^{\mathcal{P}_S}(b) & \Rightarrow \{y | y \leq a\} \subseteq \{y | y \leq b\} \\ & \Rightarrow \forall y \bullet y \leq a \Rightarrow y \leq b \\ & \Rightarrow a \leq a \Rightarrow a \leq b \end{aligned}$$

PROOF II Given $a, b \in \alpha Q$, we have $(f^{\mathcal{P}_S}(b))_L = \{y | b \leq y\}$ and $(f^{\mathcal{P}_S}(b))_H = \{y | y \leq b\}$. If $a \leq b$ then $a \in (f^{\mathcal{P}_S}(b))_H$ and by transitivity $x \leq a \Rightarrow x \in (f^{\mathcal{P}_S}(b))_H$, which implies that

$$\{y | y \leq a\} \subseteq \{y | y \leq b\}$$

and similarly we have that

$$a \leq b \Rightarrow \{y | a \leq y\} \supseteq \{y | y \leq b\}$$

and thus $a \leq b \Rightarrow f^{\mathcal{P}_S}(a) \leq f^{\mathcal{P}_S}(b)$ □

Lemma 27 Given the mapping from a quoset to a powerset lattice then if collections of classes are disjoint from one another in Q then so is their lowest upper bound in $\mathcal{P}\,\alpha Q$, i.e.,

$$\forall A, B \in \alpha Q \bullet A \updownarrow B \Rightarrow (\cup\{f^{\mathcal{P}}(a) | a \in A\}) \updownarrow (\cup\{f^{\mathcal{P}}(b) | b \in B\})$$

PROOF If $A \updownarrow B$, then for all $a \in A$ there is no $b \in B$ such that $b \leq a$. Thus, since $f^{\mathcal{P}}(a) = \{y | y \leq a\}$ then

$$b \in B \Rightarrow \neg b \in \cup\{f^{\mathcal{P}}(a) | a \in A\}$$

Similarly we have,

$$a \in A \Rightarrow \neg a \in \cup\{f^{\mathcal{P}}(b) | b \in B\}$$

Reflexivity of Q gives us

$$b \in B \Rightarrow b \in \cup\{f^{\mathcal{P}}(b)|b \in B\} \quad a \in A \Rightarrow a \in \cup\{f^{\mathcal{P}}(a)|a \in A\}$$

Combining this with the two previous equations imply that if $A \updownarrow B$ then there are components of $\{f^{\mathcal{P}}(a)|a \in A\}$ that are not in $\{f^{\mathcal{P}}(b)|b \in B\}$, and vice versa. Thus the lowest upper bounds are disjoint. □

Lemma 28 Given the mapping $f^{\mathcal{P}_S}$ from a quoset to symmetric powerset lattice then if collections of classes are disjoint from one another in Q then so are their lowest upper and greatest lower bounds in $\mathcal{P}_S\,\alpha Q$, i.e.,

$$\begin{aligned}\forall A, B \subseteq \alpha Q \bullet A \updownarrow B \;\Rightarrow\; & (\vee\{f^{\mathcal{P}_S}(a)|a \in A\}) \updownarrow (\vee\{f^{\mathcal{P}_S}(b)|b \in B\}) \wedge \\ & (\wedge\{f^{\mathcal{P}_S}(a)|a \in A\}) \updownarrow (\wedge\{f^{\mathcal{P}_S}(b)|b \in B\})\end{aligned}$$

PROOF We have,

$$\vee\{f^{\mathcal{P}_S}(a)|a \in A\} = (\{y|\forall a \in A \bullet a \le y\}, \{y|\exists a \in A \bullet y \le a\})$$

and from lemma 27 we know that if $A \updownarrow B$, then

$$(\vee\{f^{\mathcal{P}_S}(a)|a \in A\})_H \updownarrow (\vee\{f^{\mathcal{P}_S}(b)|b \in B\})_H$$

and it follows from the ordering relation over $\mathcal{P}_S\,\alpha Q$ that

$$A \updownarrow B \Rightarrow (\vee\{f^{\mathcal{P}_S}(a)|a \in A\}) \updownarrow (\vee\{f^{\mathcal{P}_S}(b)|b \in B\})$$

Similarly, if $A \updownarrow B$ we have

$$(\wedge\{f^{\mathcal{P}_S}(a)|a \in A\})_L \updownarrow (\wedge\{f^{\mathcal{P}_S}(b)|b \in B\})_L$$

which implies that

$$A \updownarrow B \Rightarrow (\wedge\{f^{\mathcal{P}_S}(a)|a \in A\}) \updownarrow (\wedge\{f^{\mathcal{P}_S}(b)|b \in B\})$$

□

Lemma 29 The mapping $f^{\mathcal{P}}_R$ from reflexive policy R to a group policy R_G built from a powerset lattice preserves the flows in R, i.e.,

$$\forall a, b \in \alpha R \bullet a \overset{R}{\leadsto} b \Leftrightarrow f^{\mathcal{P}}(a) \overset{R_G}{\leadsto} f^{\mathcal{P}}(b)$$

PROOF I We have by definition, that for $a, b \in \alpha R$,

$$f^{\mathcal{P}}(a) \overset{R_{\mathcal{G}}}{\leadsto} f^{\mathcal{P}}(b) \Rightarrow \exists X \in f^{\mathcal{P}}(a), Y \in f^{\mathcal{P}}(b) \bullet X \subseteq Y$$

but since $f^{\mathcal{P}}(a)$ forms an interval, then $X \in f^{\mathcal{P}}(a)$ implies $l^{\mathcal{P}}(a) \subseteq X \subseteq h^{\mathcal{P}}(a)$. Thus,

$$\begin{aligned} f^{\mathcal{P}}(a) \overset{R_{\mathcal{G}}}{\leadsto} f^{\mathcal{P}}(b) &\Rightarrow \exists X \in f^{\mathcal{P}}(a), Y \in f^{\mathcal{P}}(b) \bullet l^{\mathcal{P}}(a) \subseteq X \subseteq Y \subseteq h^{\mathcal{P}}(b) \\ &\Rightarrow l^{\mathcal{P}}(a) \subseteq h^{\mathcal{P}}(b) \end{aligned}$$

and since, $a \leq a$ holds, we have $a \in l^{\mathcal{P}}(a)$ and from above, $a \in h^{\mathcal{P}}(b)$. By the definition of $h^{\mathcal{P}}$, $a \in h^{\mathcal{P}}(b)$ implies $a \leadsto b$.

PROOF II The definition of bound order gives,

$$\forall y \bullet y \leq a \Rightarrow \forall x \bullet a \leq x \Rightarrow y \leq x$$

Pick $x = b$ in this equation, and if $a \leadsto b$ holds, then

$$\begin{aligned} & \forall y \bullet y \leq a \Rightarrow a \leadsto b \Rightarrow y \leadsto b \\ \Rightarrow\ & \{y | y \leq a\} \subseteq \{y \bullet | y \leadsto b\} \\ \Rightarrow\ & l^{\mathcal{P}}(a) \subseteq h^{\mathcal{P}}(b) \\ \Rightarrow\ & f^{\mathcal{P}}(a) \overset{R_{\mathcal{G}}}{\leadsto} f^{\mathcal{P}}(b) \end{aligned}$$

□

Lemma 30 The mapping $f^{\mathcal{P}_S \mathcal{G}}$ (definition 16) from R to the group policy $R_{\mathcal{G}}$ built from a symmetric powerset lattice preserves the flows in R, i.e.,

$$\forall a, b \in \alpha R \bullet a \overset{R}{\leadsto} b \Leftrightarrow f^{\mathcal{P}_S \mathcal{G}}(a) \overset{R_{\mathcal{G}}}{\leadsto} f^{\mathcal{P}_S \mathcal{G}}(b)$$

PROOF I We first note a property of the function $f^{\mathcal{P}_S \mathcal{G}}$ that it maps each element of αR onto and interval of $\mathcal{P}_S\, \alpha R$, i.e,

$$\forall x \in \alpha R \bullet l^{\mathcal{P}_S}(x) \leq h^{\mathcal{P}_S}(x) \tag{64}$$

This follows by applying the fact that if $a \leq b$ holds in R then $a \leadsto b$ holds, to the definition of $l^{\mathcal{P}_S}$ and $h^{\mathcal{P}_S}$.

Now applying this law to $\leadsto$ defined over $\mathcal{P}_S\, \alpha R$ we get for $a, b \in \alpha R$,

$$\begin{aligned} f^{\mathcal{P}_S \mathcal{G}}(a) \overset{R}{\leadsto} f^{\mathcal{P}_S \mathcal{G}}(b) &\Leftrightarrow \exists X \in f^{\mathcal{P}_S \mathcal{G}}(a), Y \in f_{(}^{\mathcal{P}_S \mathcal{G}} b) \bullet X \leq Y \\ &\Leftrightarrow l^{\mathcal{P}_S}(a) \leq h^{\mathcal{P}_S}(b) \end{aligned}$$

Unfolding $l^{\mathcal{P}s}$ and $h^{\mathcal{P}s}$ used above gives us,

$$\begin{aligned}
f^{\mathcal{P}s\,c}(a) \overset{R}{\leadsto} f^{\mathcal{P}s\,c}(b) &\Rightarrow \{y|a \leadsto y\} \supseteq \{y|b \leq y\} \\
&\Rightarrow \forall y \bullet b \leq y \Rightarrow a \leadsto y \\
&\Rightarrow b \leq b \Rightarrow a \leadsto b \\
&\Rightarrow a \leadsto b
\end{aligned}$$

PROOF II The definition of bound order gives,

$$\forall y \bullet (b \leq y \Rightarrow \forall x \bullet x \leadsto b \Rightarrow x \leadsto y) \tag{65}$$

Pick x in equation (65) above such that $x = a$, and assume that $a \leadsto b$. This gives,

$$\begin{aligned}
&\forall y \bullet b \leq y \Rightarrow a \leadsto b \Rightarrow a \leadsto y \\
\Rightarrow\quad &\{y|b \leq y\} \subseteq \{y|a \leadsto y\}
\end{aligned} \tag{66}$$

Similarly, we can prove using the fact that

$$\forall y \bullet y \leq a \Rightarrow \forall x \bullet a \leadsto x \Rightarrow y \leadsto x$$

and $a \leadsto b$ implies

$$\{y|y \leq a\} \subseteq \{y|y \leadsto b\} \tag{67}$$

Combining (66) and (67) gives

$$\begin{aligned}
a \leadsto b &\Rightarrow l^{\mathcal{P}s}(a) \leq h^{\mathcal{P}s}(b) \\
&\Rightarrow f^{\mathcal{P}s\,c}(a) \leadsto f^{\mathcal{P}s\,c}(b)
\end{aligned}$$

the desired result. □

B Transforming a Reflexive Policy to a Symmetric Powerset

The section 4.2 considered how a reflexive policy could be transformed into a powerset lattice and appropriate group confinements to be enforced by the GCFM. This section of the appendix will show how the transformation can be made to a symmetric powerset lattice.

Given a reflexive flow policy R, construct a group confinement flow policy (section 2) built from the symmetric powerset lattice of the set of security classed defined by R. The alphabet of this lattice $R_{\mathcal{G}}$ is,

$$\alpha R_{\mathcal{G}} \doteq \{X | X \subseteq \mathcal{P}_S \alpha R\} - \{\{\}, \{\}\}$$

and has a flow relation (defintion 2) defined as $(A, B \in \alpha R_{\mathcal{G}})$,

$$A \overset{R_{\mathcal{G}}}{\rightsquigarrow} B \Leftrightarrow \exists X \in A, Y \in B \bullet X \overset{\mathcal{P}_S \alpha R}{\leq} Y$$

As with a $R_{\mathcal{G}}$ built from a powerset lattice, this flow relation has a bound order defined by the partial ordering relation $\leq$ on $R_{\mathcal{G}}$. We know from section 2 that $R_{\mathcal{G}}$ does not form a lattice, but if we consider only group confinements that form intervals on $\mathcal{P}_S \alpha R$, i.e., for each confinement A then,

$$\exists l, h \in A \bullet \forall x \in A \bullet l \leq x \wedge x \leq h$$

Then the set of all such confinements are closed over $\oplus$ and $\otimes$, and it forms a sublattice (an interval lattice[10] of $R_{\mathcal{G}}$ with $\oplus$ as lowest upper and $\otimes$ as greatest lower bound operators). Furthermore, from the above we know that this interval sublattice forms a reflexive lattice with flow relation $\rightsquigarrow$ and bound order $\leq$.

Definition 16 Define a mapping $f_R^{\mathcal{P}_S \mathcal{G}}$ from arbitrary reflexive relation $R$to this sublattice of $R_{\mathcal{G}}$ as

$$f_R^{\mathcal{P}_S \mathcal{G}} : \alpha R \rightarrow \alpha R_{\mathcal{G}} \qquad f_R^{\mathcal{P}_S \mathcal{G}}(a) \doteq \{l_R^{\mathcal{P}_S}(a), h_R^{\mathcal{P}_S}(a)\}$$

where

$$\begin{aligned} l_R^{\mathcal{P}_S}(a) &\doteq (\{b | a \overset{R}{\rightsquigarrow} b\}, \{b | b \overset{R}{\leq} a\}) \\ h_R^{\mathcal{P}_S}(a) &\doteq (\{b | a \overset{R}{\leq} b\}, \{b | b \overset{R}{\rightsquigarrow} a\} \end{aligned}$$

◇

Each component of R will be mapped to an interval of $\mathcal{P}_S \alpha R$, with bottom $l^{\mathcal{P}_S}(a)$ and top $h^{\mathcal{P}_S}(a)$. Thus $f^{\mathcal{P}_S \mathcal{G}}$ is a mapping from R to a reflexive lattice. The mapping $f^{\mathcal{P}_S}$ preserves the flows of R, i.e.,

$$\forall a, b \in \alpha R \bullet a \overset{R}{\rightsquigarrow} b \Leftrightarrow f^{\mathcal{P}_S \mathcal{G}}(a) \overset{R_{\mathcal{G}}}{\rightsquigarrow} f^{\mathcal{P}_S \mathcal{G}}(b)$$

Thus we can use $f^{\mathcal{P}s\mathcal{G}}(a)$ in $R_{\mathcal{G}}$ for any class a drawn from R with no detrimental effect on flows.

In the same way that a group of classes $A \subseteq \alpha R$ gets transformed to the group $\cup\{f^{\mathcal{P}}(a)|a \in A\}$ so that it can be modelled the GCFM enforcing a powerset lattice, it can be mapped to the group $\cup\{f^{\mathcal{P}s\mathcal{G}}(a)|a \in A\}$ to be modelled in the GCFM enforcing the symmetric powerset lattice.

Example 21 A private hospital information system processes information of class **records** (medical history); **treatment** (given to patients); **accounts** (for patients); **director** (shareholder information); and **management**. How information may flow between these different classes is described by the reflexive relation in figure 9. Note how **treatment** information is allowed flow to **records** or **management**, but for confidentiality reasons, cannot flow to class **director**. Similarly, **accounts** information is not allowed flow to **records** (for profitability reasons). Management is allowed coordinate all this information given these constraints. This reflexive relation can be transformed into the reflexive lattice $R_{\mathcal{G}}$ using the mapping described in table 6. Observe from this table that if information of class **treatment** and

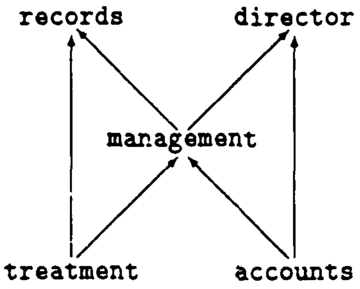


Figure 9: Flow policy HOSPITAL

x	$f^{\mathcal{P}s\mathcal{G}}(x)$
records	{({r},{tr}),({r},{tmr})}
director	{({d},{da}),({d},{dma})}
management	{({mrd},{m}),({m},{tma})}
treatment	{({tmr},{t},{tr},{t})}
accounts	{({amd},{a}),({ad},{a})}

Table 6: Mapping from $R =$ HOSPITAL to $R_{\mathcal{G}}$

accounts are combined then their lowest upper bound in $R_{\mathcal{G}}$ is not $f^{\mathcal{P}_S\mathcal{G}}(\mathtt{management})$, but $\{(\{\mathtt{m}\},\{\mathtt{ta}\}),(\{\},\{\mathtt{at}\})\}$, which may flow to class **management**, but not to **records** nor **director**.

A consultant in this hospital might be allowed sink and/or source information of class **treatment** and **records**, and thus is confined to the group $\{\mathtt{treatment},\ \mathtt{records}\}$. In the GCFM this corresponds to a binding

$$\begin{aligned} f^{\mathcal{P}_S\mathcal{G}}(\mathtt{treatment}) \cup f^{\mathcal{P}_S\mathcal{G}}(\mathtt{records}) &= \{(\{\mathtt{tmr}\},\{\mathtt{t}\}),(\{\mathtt{tr}\},\{\mathtt{t}\}),(\{\mathtt{r}\},\{\mathtt{tr}\}),(\{\mathtt{r}\},\{\mathtt{tmr}\})\} \\ &= \{(\{\mathtt{tmr}\},\{\mathtt{t}\}),(\{\mathtt{r}\},\{\mathtt{tmr}\})\} \end{aligned}$$

The hospital administrator might be bound to $\{\mathtt{management}\}$. Note how the consultant and administrator are bound to groups of 'classes' from the (symmetric powerset) lattice $\mathcal{P}_S\,\alpha\mathtt{HOSPITAL}$ which is enforced by GCFM. These classes can be thought of as representing the sources/sinks that they are permitted to make. For example, class $(\{\mathtt{mrd}\},\{\mathtt{m}\})$ means that the administrator can source accounting and record information (the $\{\mathtt{mrd}\}$ part), but in this case cannot sink accounts or treatment information (the $\{\mathtt{m}\}$ part); the administrator may also sink accounts and treatment information since $(\{\mathtt{m}\},\{\mathtt{tma}\})$ is in his group confinement, but in this case cannot source records or director information (ensures non-transitivity). Note how class **management** does not include element $(\{\mathtt{tm}\},\{\mathtt{md}\})$, ensuring that information does not flow from treatment to directors.

This range of possible classes representing **management** works neatly within the MAC model described earlier. An administrator starts our with the potential to access everything. However as he makes accesses, certain classes will be removed from his group confinement to ensure that he cannot violate the flow policy. For example, if the administrator chooses to read treatment information, he may no longer forward information to directors (but may still forward to records). We shall see further examples of these kinds of aggregate policies in the next section. $\triangle$

Thus, an arbitrary reflexive flow policy can be transformed and enforced by the GCFM. If the policy is transitive, then each component a of the policy αR will map to a singleton set $\{f^{\mathcal{P}_S\mathcal{G}}(a)\}$ in $R_{\mathcal{G}}$, since $l^{\mathcal{P}_S}(a) = h^{\mathcal{P}_S}(a)$. If the policy is not transitive, then certain components of αR will map to a group of classes from $\mathcal{P}_S\,\alpha R$ (in fact an interval, since $l^{\mathcal{P}_S}(a) \leq h^{\mathcal{P}_S}(a)$).

Example 22 Consider a market analysis database that contains information about banks Bank-x and Bank-y, and oil companies Oil-z and Oil-w. There are two conflict of interest

s	$f(s)$
bank-x	$\{(\{\mathbf{xzy}\},\{\mathbf{x}\}),(\{\mathbf{x}\},\{\mathbf{xzw}\})\}$
bank-y	$\{(\{\mathbf{yzw}\},\{\mathbf{y}\})(\{\mathbf{y}\},\{\mathbf{yzw}\})\}$
oil-z	$\{(\{\mathbf{xyz}\},\{\mathbf{z}\}),(\{\mathbf{z}\},\{\mathbf{xyz}\})\}$
oil-w	$\{(\{\mathbf{xyw}\},\{\mathbf{w}\}),(\{\mathbf{w}\},\{\mathbf{xyw}\})\}$

Table 7: Mapping for **BANKS** ⊔ **OIL**

classes **BANKS** and **OIL**. Within conflict class **BANKS** there are two kinds of information (datasets) **bank-x** and **bank-y**, corresponding to the class of information held by Bank-x and Bank-y respectively. The Chinese wall policy insists that these classes are disjoint, i.e., information about one bank is not allowed flow to another bank. Thus the conflict of interest class **BANKS** can be thought of as describing a flow policy with alphabet {**bank-x**, **bank-y**} and relations **bank-x** $\rightsquigarrow$ **bank-x**, **bank-y** $\rightsquigarrow$ **bank-y**. Conflict policy **OIL** has a similar definition, with alphabet {**oil-z**, **oil-w**} and classes **oil-z** and **oil-w** disjoint.

Now we must define how these two policies can be composed. Flows are possible between the components of the conflict policies so long as they do not violate the relations within them. Therefore, the overall flow policy can be described by the join[10] of **BANKS** and **OIL**. Policy join (⊔ of policies *C1* and *C2* has alphabet $(\alpha C1 \cup \alpha C2)$ and its flow relation is defined as $(a, b \in \alpha(C1 \sqcup C2))$,

$$\begin{aligned} a \overset{C1 \sqcup C2}{\rightsquigarrow} b \quad &\doteq \quad (a, b \in \alpha C1 \Rightarrow a \overset{C1}{\rightsquigarrow} b) \wedge \\ &\quad\;\; (a, b \in \alpha C2 \Rightarrow a \overset{C2}{\rightsquigarrow} b) \end{aligned}$$

Table 7 gives the mapping of this policy to the group lattice built from the symmetric powerset lattice $\mathcal{P}_S\{\mathtt{x,y,z,w}\}$, where **bank-x** is abbreviated to **x**, and similarly for the other classes. However, on a closer inspection of policy **BANKS** ⊔ **OIL** we discover that it is too general for our purposes here: it (correctly by its definition) permits flows from **bank-x** to **oil-z** and from **bank-y** to **oil-z** and also from **bank-x** ⊕ **bank-y** to **oil-z**, which is not desirable. Thus, we need to supplement the definition of join with a definition of how the aggregates of classes may flow in the joined policy. In the case of **oil-z** we wish to prevent it from sourcing/sinking aggregate **bank-x** ⊕ **bank-y**. Therefore given that,

$$f(\mathbf{bank\text{-}x}) \oplus f(\mathbf{bank\text{-}y}) = \{(\{\mathbf{zw}\},\{\mathbf{xy}\}),(\{\},\{\mathbf{xyzw}\})\}$$

define a class **oil-z'** as[8],

$$\begin{aligned}\texttt{oil-z'} &= f(\texttt{oil-z}) - (f(\texttt{bank-x}) \oplus f(\texttt{bank-y})) \\ &= \{(\{xyz\},\{z\}),(\{z\},\{xz\}),(\{z\},\{yz\})\}\end{aligned}$$

Note how $(\{z\},\{xyz\})$ is no longer a member of **oil-z'**. The other classes can be similarly redefined so as to constrain flows of (conflicting) aggregates,

$$\begin{aligned}\texttt{bank-x'} &= f(\texttt{bank-x}) - (f(\texttt{oil-z}) \oplus f(\texttt{oil-w})) \\ &= \{(\{xzw\},\{x\}),(\{x\},\{xz\}),(\{x\},\{xw\})\} \\ \texttt{bank-y'} &= f(\texttt{bank-y}) - (f(\texttt{oil-z}) \oplus f(\texttt{oil-w})) \\ &= \{(\{yzw\},\{y\}),(\{y\},\{yz\}),(\{y\},\{yw\})\} \\ \texttt{oil-w'} &= f(\texttt{oil-w}) - (f(\texttt{bank-x}) \oplus f(\texttt{bank-y})) \\ &= \{(\{xyw\},\{w\}),(\{w\},\{xw\}),(\{w\},\{yw\})\}\end{aligned}$$

Now, under this flow policy information is permitted to flow between the classes of different conflict policies. For example, **bank-x'** and **bank-y'** may flow to **oil-z'**, but their aggregate **bank-x'** $\oplus$ **bank-y'** may not.

We could make an additional restriction about information of class **oil-z** being permitted to sink **bank-x** or **bank-y** information, but not both, i.e., remove **bank-x** $\otimes$ **bank-y**. This gives,

$$\begin{aligned}f(\texttt{oil-z}'') &= f(\texttt{oil-z'}) - f(\texttt{bank-x}) \otimes f(\texttt{bank-y}) \\ &= \{(\{xz\},\{z\}),(\{yz\},\{z\}),(\{z\},\{xz\}),(\{z\},\{yz\})\}\end{aligned}$$

This is analagous to the requirement that once a user has written to one banking file, he cannot write to other conflicting banking files. In section 4.2 of this report, the group policy is constructed from a normal powerset lattice, and as was illustrated in section 4.1.1, cannot effectively distinguish between the lower bounds of **bank-x** and **bank-y** or **bank-x** and **oil-z**. Therefore, if we wish to express this kind of constraint on the policy a symmetric powerset lattice must be used.. Note however, that $\nabla(f(\texttt{oil-z}''))$ does not hold, and thus such a policy could not be captured by the SMM system model. However, as has already been noted, this policy is a class of integrity policy and could be enforced as such.

[8] Note that the set difference operator is defined on the largest groups in the equivalence classes of its operands.

Any information about Bank-x stored in the database will have group confinement **bank-x'**; information about Bank-y will have confinement **bank-y'**, etc. A user of the database will be confined to ($\mathtt{bank\text{-}x'} \cup \mathtt{bank\text{-}y'} \cup \mathtt{oil\text{-}z'} \cup \mathtt{oil\text{-}w'}$), allowing access to every individual item of company information but not to conflicting aggregates.

Consider a database with entries X, Y, Z and W confined as

$$\underline{X} = \mathtt{bank\text{-}x'} \qquad \underline{Z} = \mathtt{oil\text{-}z'}$$
$$\underline{Y} = \mathtt{bank\text{-}y'} \qquad \underline{W} = \mathtt{oil\text{-}w'}$$

and a user E with confinement ($\mathtt{bank\text{-}x'} \cup \mathtt{bank\text{-}y'} \cup \mathtt{oil\text{-}z'} \cup \mathtt{oil\text{-}w'}$). A system with flows $\{X, Z\} \triangleright U$ is secure since

$$\underline{X} \oplus \underline{Z} = \{(\mathtt{xz}\}, \{\mathtt{xz}\}), (\{\}, \{\mathtt{xyzw}\})\} \leadsto \underline{E}$$

Note how $\underline{X} \oplus \underline{Z}$ cannot flow to file Y. A system with flow $\{X, Y\} \triangleright U$ is not secure since

$$\underline{X} \oplus \underline{Y} = \{(\{\mathtt{zw}\}, \{\mathtt{xy}\}), (\{\}, \{\mathtt{xyzw}\})\}$$

may not flow to $\underline{E}$ since there is no component of $\underline{E}$ that contains $(\{\mathtt{zw}\}, \mathtt{xy})$. △

If the reflexive policy was implemented using a powerset lattice then there is a implementation for policy join that is defined in terms of the powerset[9]. We need a similar implementation for policy join where the policy is built using a symmetric powerset lattice.

References

[1] D.E. Bell and L.J. LaPadula. *Secure Computer Systems: Mathematical Foundations and Model.* Technical Report M74-244, The MITRE Corporation, Bedford, MA., October 1974.

[2] D.E. Bell. Security Policy Modeling for the Next-Generation Packet Switch. In *1988 IEEE Symposium on Security and Privacy*, pages 212–216, IEEE Computer Society, Oakland, CA., April 1988.

[3] K.J. Biba. *Integrity Considerations for Secure Computer Systems.* Technical Report MTR-3153, The Mitre Corporation, Bedford, MA., April 1977.

[4] G. Birkhoff. *Lattice Theory.* Volume XXV of *American Mathematical Society Colloqium Publications*, American Mathematical Society, 3 edition, 1967.

[5] D.F. Brewer and M.J. Nash. The Chinese Wall Security Policy. *Proceedings 1989 Symposium on security and Privacy*, pages 206–258, IEEE Computer Society, Oakland, CA., May 1989.

[6] D.E. Denning. A Lattice Model of Secure Information Flow. *Communications of the ACM*, 19(5):236–243, May 1976.

[7] D.E. Denning. *On the Derivation of Lattice Structured Information Flow Policies.* Technical Report CSD TR180, Purdue University, 1976.

[8] D.E. Denning and P.J. Denning. Certification of programs for secure information flow. *Communications of the ACM*, 20(7):504–513, July 1977.

[9] S.N. Foley. *A Theory and Model for Secure Information Flow.* PhD thesis, National University of Ireland, 1988.

[10] S.N. Foley. A Model for Secure Information Flow In *1989 IEEE Symposium on Security and Privacy*, IEEE Computer Society, Oakland, May, 1989.

[11] C. L. Harrold. *Formal Specification of a Secure Document Control System for SMITE* RSRE Report No. 88002, Malvern, Worcs., 1988.

[12] T.Y. Lin. Chinese Wall Security Policy – An aggressive Model. In *Proceedings of the 5th Aerospace Symposium on Computer Security and Privacy,* 1989.

[13] G. H. MacEwen. The Design for a Secure System Based on Program Analysis *IEEE Trans. Soft. Eng.*, SE9(3), pp 289–298, May 1983.

[14] J. McLean. The Algebra of Security. In *1988 IEEE Symposium on Security and Privacy*, pages 2–7, IEEE Computer Society, Oakland, CA., April 1988.

[15] C. Meadows. Extending the Brewer-Nash Model to a Multi-level Context. To appear in *1990 IEEE Symposium on Security and Privacy*, IEEE Computer Society, Oakland, CA., May 1990.

[16] P.F. Terry and S.R. Wisemman. A 'New' Security Policy Model. In *1989 IEEE Symposium on Security and Privacy*, Oakland CA., May 1989.

[17] C. Weissman. Security Controls in the ADEPT-50 Time Sharing System. In *Proceedings of the 1969 AFIPS Fall Joint Computer Conference*, pages 119–133, AFIPS Press, Arlington, Va, 1969.

REPORT DOCUMENTATION PAGE

DRIC Reference Number (if known) ....

Overall security classification of sheet ....Unclassified..

(As far as possible this sheet should contain only unclassified information. If it is necessary to enter classified information, the field concerned must be marked to indicate the classification eg (R), (C) or (S).

Originators Reference/Report No. REPORT 90005	Month APRIL	Year 1990

Originators Name and Location

RSRE, St Andrews Road
Malvern, Worcs WR14 3PS

Monitoring Agency Name and Location

Title

LATTICES FOR SECURITY POLICIES

Report Security Classification UNCLASSIFIED	Title Classification (U, R, C or S) U

Foreign Language Title (in the case of translations)

Conference Details

Agency Reference	Contract Number and Period
Project Number	Other References
Authors FOLEY, S N	Pagination and Ref 87

Abstract

This report lays the foundation for a new model and approach for secure information flow. The model is driven by lattice based information flow policy. which describes the permitted dissemination of information in the system. System entities are allowed to handle different classes of information from the flow policy, and information is permitted to flow between entities so long as they do not violate the flow policy.

With this conceptually simple notion of security we can describe many interesting security policies, for example, traditional multi-level policies, aggregation policies, and chinese walls. Details are given on how secure systems based on the model can be implemented in practice. We also examine how other types of security policies such as integrity and separation of duty can be defined in terms of lattice based policies.

Abstract Classification (U,R,C or S)
U

Descriptors

Distribution Statement (Enter any limitations on the distribution of the document)

UNLIMITED